Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Zachovanie mentálnej mapy hrán pri interakcii s grafom

Diplomová práca

Bratislava, 2013

Bc. Martin Ďuriš

Univerzita Komenského v Bratislave

Fakulta matematiky, fyziky a informatiky

Zachovanie mentálnej mapy hrán pri interakcii s grafom

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Fakulta Matematiky, Fyziky a Informatiky
Školiteľ:	RNDr. Jana Katrenuaková PhD.

Bratislava, 2013

Bc. Martin Ďuriš

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Martin Ďuriš
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Zachovanie mentálnej mapy hrán pri interakcii s grafom

Cieľ: V oblasti dynamického kreslenia grafov sa ukazuje, že oveľa dôležitejšie, ako statické estetické kritériá je zachovanie mentálnej mapy používateľa - t.j. nie veľké zmeny vykreslenia pri úpravách v grafe. Mentálna mapa sa však väčšinou používa v súvislosti s polohami vrcholov. V prípade, že tieto sú pevné alebo zadané používateľom, potrebujeme vykresľovať iba hrany. Ich vykresľovanie je však závislé od aktuálne zobrazných vrcholov a môže sa aj pridaním alebo zrušením jedného vrchola výrazne zmeniť. Cieľom práce je rozšíriť pojem mentálnej mapy aj na prípad kreslenia hrán. Analýzou algoritmov pre kreslenie hrán následne určíme, aké sú potrebné modifikácie na to, aby sa im takto definovaný model mentálnej mapy podarilo zachovávať.

Vedúci: RNDr. Jana Katreniaková, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: doc. RNDr. Daniel Olejár, PhD.

Dátum zadania: 14.10.2011

Dátum schválenia: 02.11.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Chcel by som poďakovať predovšetkým svojej vedúcej diplomovej práce Jane Katreniakovej, ktorá mi venovala množstvo času tak ako aj mnoho vecných odborných i praktických rád. Poďakovať chcem aj svojej rodine, priateľom a blízkym za ich morálnu podporu pri tvorení tejto práci.

Abstrakt

Pri vykresľovaní grafu môže interakcia s grafom spôsobit' vykreslenie výrazne odlišného grafu ako bol pred samotnou interakciou. Známe statické kritéria na vizualizáciu grafov pri interakcia s grafom nie sú postačujúce na to, aby používateľ vedel plnohodnotne vnímať prebiehajúce zmeny v grafe, špeciálne hrán a používateľ tak stráca mentálnu mapu hrán. Práve dynamické kritéria na kreslenie zhoršujú schopnosť udržania mentálnej mapy hrán ale aj vrcholov. Doteraz známe práce sa venovali zachovávaniu mentálnej mapy vrcholov. V našej práci sme na zachovanie mentálnej mapy hrán navrhli dva modely. V pripravenej aplikácii sme skúmali, či ich aplikovanie zlepší zachovávanie mentálnej mapy hrán.

Kľúčové slová : Mentálna mapa, zachovávanie mentálnej mapy, graf, kresliace algoritmy

Abstract

During drawing of graph interaction with graph can cause drawing of markedly different graph than was before interaction. Known static criteria for graph visualization are not sufficient for user understanding of graph changes, that occur during interaction - especially edges and user is loosing mental map of edges. Dynamic criteria for drawing make preserving mental map of edges as vertices worse. Up to now know articles has focused only on preserving mental map of vertices. In our work we are suggesting two models for maintaining mental map of vertices. For testing purposes we have developed application. In this application we check if preserving mental map of edges has optimize graph drawing.

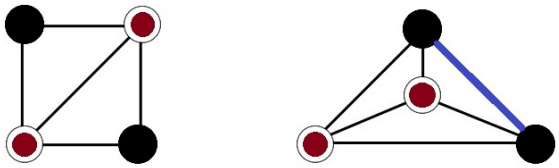
Keywords : Mental map, preserving of mental map, graph, drawing algorithms

Obsah

Úvod

Interakcia s grafom, či už v podobne zmazania alebo pridania vrchola, pohybom s existujúcim vrcholom či zmeny v množine hrán grafu, môže spôsobiť vykreslenie na prvý pohľad odlišného grafu ako bol pred samotnou interakciou. Väčšina vykresľovacích algoritmov sa snaží dosiahnuť, aby jeho výsledkom bol vykreslený obrázok grafu, ktorý spĺňa niektoré vopred definované podmienky ako napr. vytvorenie planárneho grafu, neprekrývanie sa vrcholov, zoskupovanie vrcholov do klastrov - skupín, hľadanie najkratších hrán a mnoho iných.

Aj jednoduchá zmena môže pri kreslení grafu spôsobiť, že jeho nová podoba bude mať s pôvodnou len málo spoločného. Keby sme chceli spájať vrcholy s najkratšími hranami, tak po odstránení jedného vrcholu sa môže niektorá z hrán natoľko zmeniť, že v novom grafe by sme ju pri prvom pohľade vôbec nenašli. Ako ďalší príklad môže uviesť snahu o vytvorenie planárneho grafu.



Obr. 1: *Obr. A a obr. B*

Ku grafu na obrázku A pridáme jednu hranu medzi čiernymi vrcholmi. Kresliaci algoritmus by mohol vytvoriť graf ako je na obrázku B, ktorý síce patrí medzi planárne grafy, no jeho podobnosť s grafom A z ktorého vznikol je veľmi malá.

Cieľom tejto práce bude navrhnúť spôsob, ako by sa dala kresliť interakcia na grafe tak, aby sme veľmi neporušili jej spätosť s grafom pred interakciou. Konkrétne sa zameriame na kreslenie hrán. Udržiavanie takejto nadväznosti, ako sa ukázala v niektorých predchádzajúcich prácach, môže značne ovplyvniť používateľovo pochopenie grafu a zmien v ňom.

Podobný problém bol už riešený pri interakcii s vrcholmi na dynamických grafoch – teda grafoch, ktoré sa v priebehu času mohli meniť. Práce sa zameriavali predovšetkým na obmedzenie zmien pri vykresľovaní vrcholov. Okrem nových vykreslovacích algoritmoch sa zaviedli pre grafy aj špeciálne nástroje, takzvané mentálne mapy, ktorých použitie malo značný vplyv na výsledok.

Kapitola 1

Vizualizácia grafu

1.1 Vykresľovacie algoritmy

Vykresľovacie algoritmy [1] [2] [3], ktoré sa používajú pri vizualizáciach by sme mohli rozdeliť do niekoľkých tried. Prvé vykreslovacie algoritmy boli silovo zamerané (force - directed drawing algorithm) - teda algoritmy, ktoré využívali fyzikálne zákony. Vrcholy v takomto grafe si môžeme predstaviť ako predmety / objekty, ktoré sú pospájane pružinami - teda hranami. Vrcholom sú nastavené počiatočné polohy a následne sa v niekoľkých iteráciách vypočíta ich výsledná poloha. Na každý vrchol pôsobia príťažlivé sily a odpudivé sily. V jednotlivých iteráciách sa tieto sily aplikujú a mení sa poloha vrcholov.

V roku 1963 bol vytvorený prvý takýto algoritmus[4]. Navrhol ho matematik Tutte na bariocentrickej metóde. Riešením sústavy rovníc tento algoritmus zaručoval priamočiare, neprekrývajúce sa hrany v grafe (musel byť ale 3-súvislý planárny). Medzi tieto algoritmy môžeme spomenúť aj Eadesov algoritmus (1984) [5]. Tieto algoritmy sa ďalej skúmali a skúšali ich rôzne obmeny.

V roku 1999[6] sa začala venovať pozornosť grafom s počtom vrcholov viac ako 1000. Predstavený bol algoritmus od Hadana a Harela. Tento algoritmus pri prvotnom prerozdelení vrcholov úmyselne porušoval niektoré detaily grafu (nesnažil

sa o planaritu grafu a pod.). Dostal takto hrubý náčrt grafu, ako asi môže vyzerať po vykreslení. Vynechané detaily sú v nasledujúcich krokoch postupne zapojené pri prepočítavaní nových polôh vrcholov v grafe.

Počet vrcholov sa aj naďalej zväčšoval[7] a nové algoritmy už prestali zahŕňať do svojich podmienok presné číslo, pre koľko vrcholov sú obmedzované.

Všetky tieto algoritmy pracovali v pre nás bežnom Euklidovom priestore. Predstavených bolo ale aj niekoľko myšlienok, ako by sa dalo získať vykreslenie grafu aj v inom priestore. V hyperbolickom priestore[8] sa dá vypočítať vykreslenie kompletného stromu s uniformnou dĺžkou hrán a uniformne distribuovanými vrcholmi. Koburov a Wampler[9] popísali spôsob, ako zas transformovať Euklidovu geometriu do Riemannovej geometrie.

Všetky tieto algoritmy ale rátali s jedným statickým vykreslením grafu. Dynamika v grafoch - ich zmena v čase - doteraz nebola braná do úvahy.

1.2 Mentálna mapa

Pri grafoch, ktoré sa menia v čase – či už vplyvom vývoja grafu alebo vplyvom človeka - je dôležité kontinuálne porozumenie. Grafy totiž môžeme použiť na vizualizáciu mnohých problémov a pochopenie tohto grafu (ako sú napr. spôsob spájania vrcholov, topológia, zvýraznené dôležité časti grafu a pod.) je kľúčové. Príkladom sú siete, grafy s veľkým počtom vrcholov a hrán, ktoré v čase simulujú veľa krát skutočné objekty zo sveta (napr. vyťaženie elektrickej siete).

Je teda dôležité kresliť esteticky vyhovujúce grafy (planárne grafy, neprekrývajúce sa vrcholy ...) - tieto kritéria označujeme ako **statické**. Keby sme chceli hodnotiť vykreslenie grafu, mohli by sme jednoducho zrátať počet prekrývajúcich sa vrcholov, križujúcich sa hrán a mnoho iných statických kritérií. Čim lepšie hodnotenie dosiahol graf, tým je graf 'krajší' - estetickejší.

Neskôr sa pri vykreslovaní grafov pridala **interakcia**. Interakcia s grafom je zmena polohy vrcholu grafu, zmazanie / pridanie vrcholu alebo odobratie / pridanie

hrany. Túto interakciu vykonáva často používateľ. V oblasti komplexných sieti[10] sa používajú grafy na simuláciu rôznych štruktúr (napr. elektrická sieť). Táto štruktúra sa v čase mení (napr. elektrická sieť ma rôzne využitie cez deň a cez noc) a dochádza tak k zmene grafu - teda k interakcií.

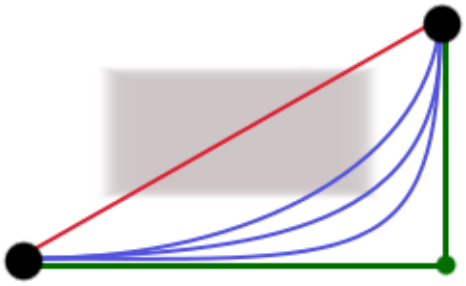
Druhú triedu kritérií môžme označiť ako **dynamické**. Do tejto triedy sa budú radiť všetky požiadavky na estetické vykreslenie grafu pri interakcií (či už používateľom, alebo inými udalosťami, ktoré ovplyvňujú polohu vrcholov a hrán grafu). Interakcia s grafom je činnosť, ktorá sa deje v čase (ako napríklad sieť znázorňujúca vyťaženie elektrickej siete za obdobie jedného mesiaca). Keby sme teda chceli zaviesť, tak ako pri statických kritériách, spôsob hodnotenia novo vykresleného grafu, musíme rátať s tým, že hodnotenie nie je iba závislé iba na momentálnom vykreslení, ale aj od predchádzajúceho vykreslenia. Pre riešenie takejto situácie bol zavedený pojem mentálna mapa, ktorá dokáže zachytiť graf a jeho zmeny v čase. Zachovávanie tejto mentálnej mapy by malo spôsobiť vykresľovanie grafu tak, aby používateľ dokázal vždy zaregistrovať všetky zmeny v grafe.

Mentálna mapa je informácia, ktorú si uchováva používateľ pri tom, ako sa graf mení pri zmenách vo vykreslení. S každou zmenou, ktorá ovplyvnila vykreslenie grafu, sa mentálna mapa aktualizuje. Úlohou tejto mapy je zachytiť schému grafu, teda rozloženie jeho vrcholov a hrán. Jednoduchý príklad mentálnej mapy je zapamätanie si polohy každého vrcholu po vykreslení.

Zachovávanie mentálnej mapy je proces, kedy sa porovnáva mentálna mapa pred a po interakcií s grafom. Tento proces rozhoduje, či porovnávané mentálne mapy, sú navzájom podobné a keď je to potrebné, usmerňuje vykresľovací algoritmus. Ak sú identické, alebo sa odlišujú len v tolerovanej miere, hovoríme, že mentálna mapa sa zachováva. V takomto prípade návrh na vykreslenie cesty je od kresliaceho algoritmu je akceptovaný. V opačnom prípade, musí dôjsť k úprave vykreslenia cesty. Výstupom tohto procesu môže byť aj hodnota kvality - ako veľmi dobre sa mapy zachovávajú.

Proces zachovania mentálnej mapy môže byť napríklad kontrolovanie, či počet vrcholov ktoré zmenili svoju polohu, nepresiahol stanovenú konštantu. Obrázok *1.1* ilustruje ukážku, ako vyzerá vykreslenie hrán grafu po interakcií pri zachovávaní

mentálnej mapy hrán.



Obr. 1.1: *Zelená je cesta pred interakciou (obchádzala prekážku). Červená je navrhovaná cesta od kresliaceho algoritmu. Táto cesta by sa ale dala nakresliť aj iná̋č - tak, aby sa oveľa viac podobala pôvodnej ceste a mentálna mapa hrany sa zachovávala. Modré sú možné výsledne cesty, ktoré považujeme za správne.*

Doterajšie práce[11] o mentálnych mapách a ich zachovaní sa zameriavali predovšetkým na zachovávanie mentálnej mapy vrcholov. Ako sa ukázalo, niektoré riešenia napr. obmedzenie zmeny polohy vrcholov[12]) neviedli k očakávaným výsledkom a pochopenie zmien v grafe sa dokonca aj zhoršovalo. Táto práca rieši problém reprezentácie mentálnej mapy hrán a ponúka dva spôsoby, ako sa môže zachovávať. Pri návrhu procesu zachovania mentálnej mapy sa nekládli žiadne podmienky na kresliaci algoritmus.

1.3 Kreslenie grafu s mentálnou mapou

Graf by sa pri interakcií mal čo najmenej meniť, aby sa tak zachovalo jeho kontinuálne pochopenie - teda mentálna mapa. To, ako sa bude mentálna mapa používateľa meniť pri práci s grafom, je závislé predovšetkým na kresliacom algoritme. Kresliaci algoritmus totiž mení vykreslenie grafu a teda má priami vplyv na mentálnu mapu. Kresliaci algoritmus dostane za úlohu vykresliť graf a mentálna mapa používateľa sa zmení. Algoritmus pre vykresľovanie grafu a mentálnu mapu preto chápeme ako dva úzko spolupracujúce objekty.

Kresliaci algoritmus ponúkne nové možné vykreslenie grafu, v ktorých sú už vykonané zmeny. Z tychto zmien si nami navrhované procesy na zachovanie mentálnej

mapy budú všímať predovšetkým zmenu ciest medzi vrcholmi. Cesta medzi vrcholmi sa dá jednoducho popísať ako zoznam bodov, kde cesta mení smer[1]. Tieto cesty ale musia zachovávať mentálna mapu hrán - overí sa, či nedošlo k veľmi veľkým zmenám v grafe, aby sa neporušilo porozumenie grafu.

Aby nová cesta nebola veľmi odlišná (aby sa zachovala mentálna mapa), môže sa požiadať kresliaci algoritmus o ďalšiu cestu – pričom sa mu dodajú dodatočné parametre (ako napr. body, cez ktoré musí prechádzať, presne vymedzená oblasť v ktorej sa cesta môže nachádzať a pod.). Mentálna mapa môže požiadať aj o viacero ciest a vybrať z nich tú najlepšiu, prípadne iba o nájdenie časti cesty.

Ako príklad praktického využitia mentálnych máp a snahy zachovať podobnosť hrán pri interakcií na grafoch, môžeme uviesť rôzne programy pre Objektové modelovanie a analýzu (UML diagramy), navrhovanie databáz, VLSI (proces tvorby integrovaných obvodov) alebo rôzne príklady z oblasti problematiky sieti.

1.4 Mentálne mapy vrcholov

Pre zachovávanie mentálnej mapy vrcholov už bolo popísaných niekoľko spôsobov. Niektoré z matematických modelov používateľovej mentálnej mapy popíšeme v tejto časti. Zachovanie mentálnej mapy sa dá definovať ako ekvivalenciu medzi dvoma množinami objektov - mentálných máp vrcholov. Jedná mentálna mapa obsahuje pôvodné vrcholy pred interakciou a druhá body, ktoré vznikli po interakcií s grafom. Hovoríme, že mentálna mapa medzi takýmito dvoma množina je zachovaná vtedy, ak podmienky resp. obmedzenia, ktorými je popísaná mentálna mapa používateľov, sú splnené po interakcií.

Jedným z najzákladnejších modelov mentálnej mapy a jej zachovávania je ortogonálne zoraďovanie. Medzi dvoma vrcholmi p a q si mentálna mapa uchová ich smerovanie - na ktorej svetovej stane vrchol q od vrcholu p leží. Väčšinou si táto mentálna mapa vystačí s ôsmimi stranami (sever, severo-východ, východ, juho-východ atď.). Smerovanie si zapamätáme aj z 'opačnej strany' od vrcholu q do vrcholu p a

[1]Predpokladáme, že cesty sú iba úsečky. Krivky v našej práci neberieme do úvahy.

pre všetky kombinácie vrcholov. Vznikne nám matica (veľkosť je V - počet vrcholov graf), v ktorej sú tieto smery zaznamenané. Interakcia zachovaná mentálnu mapu, ak sa táto matica nezmenila len v tolerovanom rozmedzí.

Ďalší intuitívny model je sledovanie klastrov. Klaster je zoskupenie vrcholov do oblasti po vykreslení. Pre klaster sa zvolí epsilón - odchýlka a pre každé dva vrcholy v klastri musí platiť $d(p,q) \leq epsilon$. Pri tomto modely bol popísaný aj koncept grafu s názvom "sphere of influence graph oblasť vplyvu grafu. Takýto graf má vrcholy ako graf, pre ktorý túto štruktúru vytvárame a hrany ak :

$$p \in S, q \in S, d(p,q) \leq min_{r \in S} d(p,r) + min_{s \in S} d(q,s)$$
$$S \textit{ je množina vrcholov grafu}$$

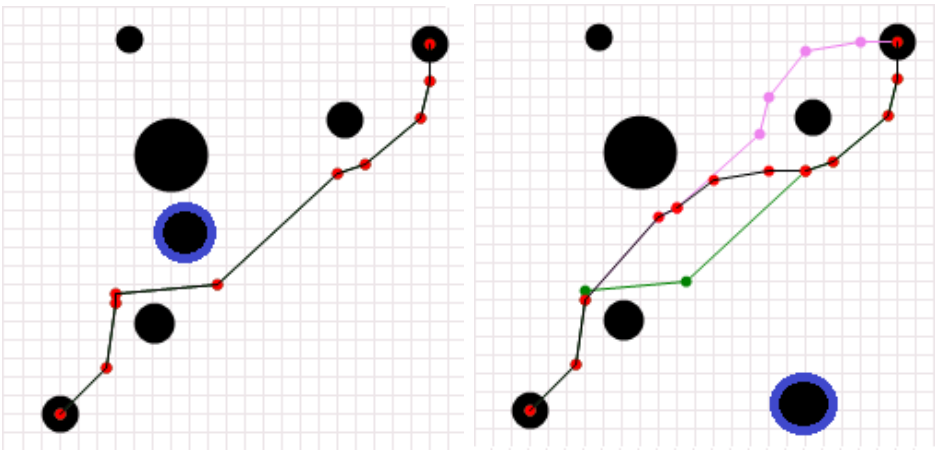
Toussaint argumentuje[11], že takáto štruktúra presne zachytáva perceptuálne štruktúry scény bodového vzoru na veľmi nízkej úrovni.

Homomorfizmus transformácií sa používal pri zachovávaní topologickej stránky vykresleného grafu. Vybrané boli niektoré operácie, ktoré zachovávajú homomorfizmus topológie a tieto jediné operácie sa mohli používať pri snahe o zmenu polohu vrcholov. Zaujímavé transformácie boli reflexia, rotácia, translácia a škálovanie. Dovolené boli aj ľubovolné iné kombinácie.

Kapitola 2

Obmedzenie nových bodov na ceste

V tejto kapitole sa venujeme modelu zachovania mentálnej mapy, kde obmedzíme počet odlišných vrcholov. Na obrázku *2.1* sú vykreslené dva obrázky. Na obrázku *A* je zvýraznený vrchol, ktorý bude pri interakcií presunutý na iné miesto. Obrázok *B* ukazuje, ako sa zmenila poloha zvýrazneného vrcholu spolu s troma cestami. Všetky tri cesty majú spoločných niekoľko bodov, neskôr menia svoj tvar. Zelená je cesta pred interakciou. Fialová je cesta, ktorú navrhuje kresliaci algoritmus po interakcií. Čierna cesta je výsledná, ktorá sa použije pri vykreslení, pri zachovávaní mentálnej mapy hrán pomocou modelu, ktorý je v tejto kapitole popísaný.



Obr. 2.1: *Obrázok A (vľavo) a obrázok B (vpravo)*

Cesta z bodu X do Y obsahuje niekoľko dôležitých bodov. Sú to body, v ktorých cesta mení smer. Tieto body vznikajú prirodzene pri tom, ako kresliaci algoritmus hľadá cestu a snaží sa pri tom splniť svoje kritéria (planarita vykresleného grafu a pod.). Pomocou tých bodov označujeme cestu :

Definícia 1. *Cesta z bodu X do bodu Y označujeme ako :* $XY_P = \{V_0, V_1, V_2, ..., V_n\}$, *kde body* V_0 *až* V_n *sú body, v ktorých cesta mení smer.*

Pri reprezentácii mentálnej mapy hrán práve tieto body zohrávajú kľúčovú rolu. Pre každú cestu sa uloží množina týchto bodov. Množina všetkých takýchto bodov pre všetkých cesty predstavujú mentálnu mapu vrcholov.

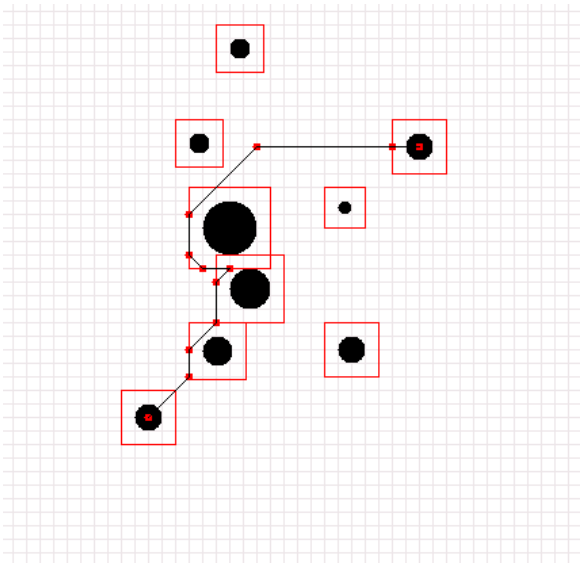
2.1 Obmedzený počet nových významných bodov na ceste

Tento model zachovania mentálnej mapy vrcholov vychádza z myšlienok a výsledkov mentálnych máp vrcholov[11]. Cieľom je pomocou konštanty (alebo sady konštánt) zamedziť vznik nových významných bodov na ceste. S každým novým významným bodom na ceste sa nová cesta mierne odlišuje od pôvodnej cesty (pred interakciou).

Prvé nájdenie cesty nie je obmedzované. Graf je prvý krát vykreslený a mentálna mapa ešte neexistuje. Pri nasledujúcej interakcií s grafom sa už každá hrana upravuje podľa modelu zachovávania mentálnej mapy. Hlavnou myšlienkou tohto modelu je dovolenie iba niekoľkých nových bodov pre každú cestu. Ak sa v grafe niektorý vrchol zmazal, pridal, alebo zmenil polohu je potrebné zistiť, ako táto zmena pôsobí na hrany.

Označenie ciest, ktoré sa v tomto modely používajú :

- Cesta pred interakciou $XY_P = \{V_0, V_1, V_2, ..., V_n\}, n \geq 0$
- Cesta, ktorú navrhuje kresliaci algoritmus : $XY_{Algo} = \{V_0^{'}, V_1^{'}, V_2^{'}, ..., V_m^{'}\}, m \geq 0$



Obr. 2.2: *Ukážka vykresleného grafu pomocou ASTAR algoritmu*

- Výsledná cesta, ktorá sa vykreslí : $XY_{Final} = \{V_0^F, V_1^F, V_2^F, ..., V_l^F, t \geq 0\}$

Keď n a m sú rovnaké a každý jeden bod je rovnaký ($\forall i V_i = V_i^{'}$), potom zmena neovplyvnila hranu (dva vrcholy sú rovnaké vtedy, ak majú X-ovú aj Y-ovú súradnicu rovnakú). Výsledná cesta XY_{Final} bude rovnaká ako XY_P. Mentálna mapa pre túto cestu bola zachovaná.

Nech $V_a^{'}$ je prvý odlišný vrchol na ceste XY_{Algo} (je odlišný ako ako vrchol V_a). Cieľom modelu zachovávania mentálnej mapy bude vytvoriť novú cestu XY_{Final} z ciest XY_P a XY_{Algo}.

Nová cesta XY_{Final} sa ale musí od pôvodnej cesty XY_P líšiť iba v maximálne K vrcholoch. Číslo K je parameter modelu a priamo ovplyvňuje, ako veľmi dovolíme zmeny pri kreslení ciest, resp. ako veľa vrcholov z novej cesty XY_{Algo} pridáme do novej cesty XY_{Final}.

Skladanie novej cesty

Identická časť cesty : Prvé vrcholy V_0 až V_{a-1} budú vo výslednej ceste identické s cestou XY_P.

$$XY_{Final} = \{V_0, V_1, V_2, ..., V_{a-1}\},\ V_0 \text{ až } V_{a-1} \in XY_P$$

Nová časť cesty : Od bodu V_a ale nastupujú nové vrcholy z cesty XY_{Algo}. Keďže je povolených K vrcholov z cesty XY_{Algo}, nové vrcholy sa do finálnej cesty doplnia z tejto cesty :

$$XY_{Final} = \{V_0, ..., V_{a-1}, V_a, ..., V_{a+K}\},\ V_0 \text{ až } V_{a-1} \in XY_P \text{ a } V_a \text{ až } V_{a+K} \in XY_{Algo}$$

Ak niektoré z vrcholov, ktoré boli pridané z cesty XY_{Algo} bol aj posledný vrchol cesty XY_P, vytváranie cesty XY_{Final} je skončené. Nová cesta sa líši v maximálne K vrcholoch a mentálna mapa cesty sa zachováva - jej hlavná podmienka, aby nová cesta mala maximálne K nových vrcholov je zjavne dodržaná.

Doplnenie záveru cesty : Ak sa ale ešte nedosiahol posledný vrchol cesty XY_P, musí sa výsledná cesta dokončiť vrcholmi z cesty XY_{Final}. K ceste XY_{Final} pridáme zvyšných niekoľko z vrcholov cesty XY_P.

$$XY_{Final} = \{V_0, ..., V_{a-1}, V_a, ..., V_{a+K}, V_{a+K+1}, ..., V_n\},\ V_0 \text{ až } V_{a-1} \in XY_P,\ V_a \text{ až } V_{a+K} \in XY_{Algo},\ V_{a+K+1} \text{ až } V_n \in XY_P$$

Medzi vrcholmi V_{a+K} a V_{a+K+1} sa ale musí dodatočne nájsť cesta. Cestu vytvorí algoritmus, ktorý vytvoril aj cestu XY_{Algo}. Na ceste XY_{Final} sa teda medzi vrcholmi V_{a+K} a V_{a+K+1} mohlo pridať niekoľko nových vrcholov. Počet nových vrcholov v hrane XY_{Final} je teda väčší počet, ako sme chceli dovoliť, no mentálnu mapu hrany môžme považovať za zachovanú (predpokladá sa, že vzdialenosť vrcholmi V_{a+K} a V_{a+K+1} zväčša nebude príliš veľká). Hlavné časti hrany XY_P sa ale vo výslednej hrane XY_{Final} nachádzajú - teda jej črty a podobnosť sú zachované.

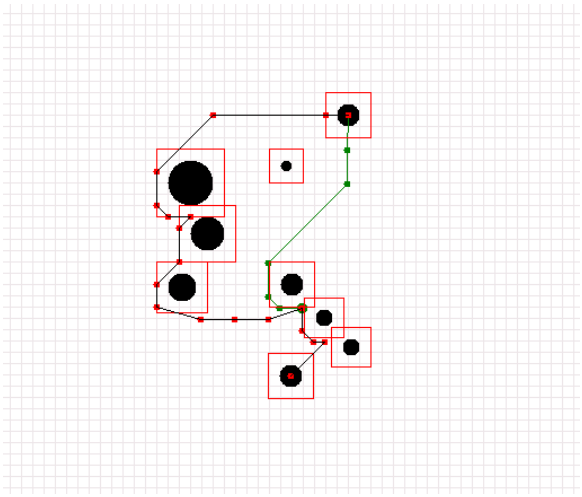
Definícia 2. *Výsledne cesty pri modely zachovávania mentálnej mapy, ktorý obmedzuje počet nových vrcholov, vznikajú ako*[1] *:*

[1]Predpokladá sa, že $m \geq n$. Ináč, ak $m<n$, cesta XY_{Algo} musí byť odlišná od XY_P - rôzny počet bodov znamená, že cesty nie sú identické. Postupovať sa môže rovnako ako pre $m \geq n$. Ak $m<(a+K)$, nemusí sa vyžadovať K nových vrcholov - doplniť stačí vrcholy $V_a, ..., V_{a+m}$

$$XY_{Final} = \{V_0, ..., V_{a-1}, V_a, ..., V_{a+K}, V_{a+K+1}, ..., V_n\},$$
$$V_0 \text{ až } V_{a-1} \in XY_P,\ V_a \text{ až } V_{a+K} \in XY_{Algo},\ V_{a+K+1} \text{ až } V_n \in XY_P \text{ ak } n > (a+k)$$

alebo

$$XY_{Final} = \{V_0, ..., V_{a-1}, V_a, ..., V_{a+l}\},$$
$$V_0 \text{ až } V_{a-1} \in XY_P,\ V_a \text{ až } V_{a+l} \in XY_{Algo},\ l \leq k \text{ a } n \leq (a+k)$$

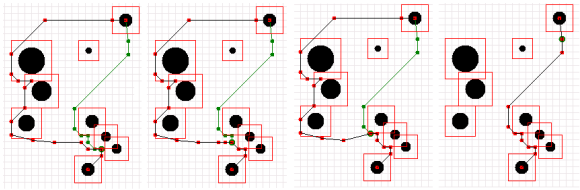


Obr. 2.3: *Ukážka grafu po zmene. Zelenou čiarou je hrana, ktorú by vykresľovací algoritmus pôvodne vykreslil*

Problém pri tomto modeli zachovávania mentálnej mapy vrcholov môže nastať ale vtedy, keď K nových vrcholov nestačí na úspešné obídenie zmeny v grafe. Situáciu si môžme predstaviť na nasledujúcom príklade. Pri posledný vrchol cesty vložíme dostatočne veľký nový vrchol (jeho veľkosť ale môžme nahradiť aj jeho správne zvolenou polohou). Nová cesta teda musí od bodu V_a obchádzať nový vrchol. Keď sme pridali K bodov z novej cesty XY_{Algo} ako ďalší sa pridal bod z cesty XY_P. Cesta medzi týmito dvoma vrcholmi sa hľadala pomocou kresliaceho algoritmu, alebo sa len priamo spojila. Pri oboch riešeniach ale výsledne vykreslenie nemuselo byť z pohľadu estetiky vôbec prijateľný krok (na tomto vloženom úseko vznikla veľmi komplikovaná cesta). Môžeme povedať, spojenie ciest XY_P a XY_{Algo} nie je estetické a navyše sa nezachováva mentálna mapa hrán.

Nasledujúca interakcia na grafe pri použití tohto modelu zachovávania mentálnej

mapy tento problém ale vyrieši. Ak totiž nastane niekde v grafe ďalšia zmena, na ľubovolnom vrchole, dostaneme novú cestu XY_{Algo2}, ktorá už ale nemá z cesty XY_P $N-K$ vrcholov, ale iba $N-2K$ vrcholov. Do cesty XY_{Algo2} oproti ceste XY_{Algo} bolo pridaných nových K vrcholov z cesty navrhovanej vykresľovacím algoritmom. Po niekoľkých takýchto 'iteráciách' sa hrana nakoniec stane tou, ktorú navrhuje vykresľovací algoritmus - estetika, ktorú nám tento algoritmus zaručoval vo vykreslenom grafe nakoniec nájdeme. Zmeny v novej ceste sú rozdelené do niekoľkých krokov a mentálna mapa sa bude kontinuálne meniť vždy len o menšie, prijateľnejšie úseky.



Obr. 2.4: *Ukážka pri veľmi malej konštante K ako hrana postupne prejde do podoby, ako ju navrhuje vykresľovací algoritmus*

Predstaviť sa dá aj menšia obmena tejto mentálnej mapy. Cestu medzi vrcholmi, kde sme dodatočne hľadali kresliacim algoritmom novú cestu, nemusíme hľadať vykresľovacím algoritmom. Tieto dva vrcholy môžeme jednoducho spojiť hranou. Takáto hrana ale mohla ľahko spôsobiť problémy z estetickej stránky vykresleného grafu. Ako sme ale už ukázali, pri ďalších vykresleniach / iteráciách sa táto hrana "nahradí" ďalšími časťami hrany, ktorú navrhol kresliaci algoritmus.

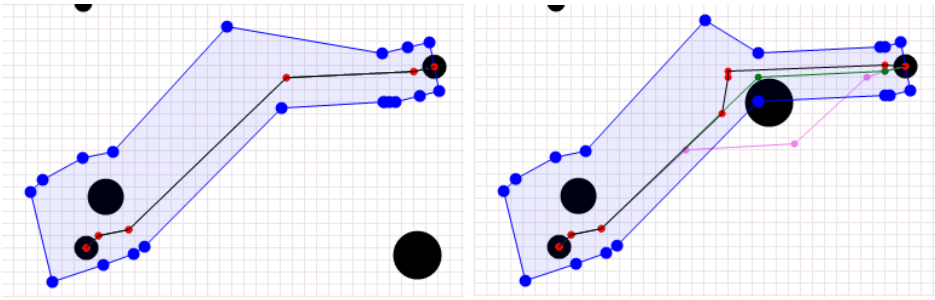
Diskusia

Tento model pristúpil k problému zachovania mentálnej mapy spôsobom, kedy zmenu vo vykreslení hrany, či už malú alebo veľkú, rozdelí do niekoľkých iterácií. Ak sa v nasledujúcich ďalších iteráciách zmena v grafe deje v inej oblasti, model je úspešný. Po niekoľkých iteráciách je používateľ prijateľným spôsobom pripravený na vykreslenie navrhovanie cesty od kresliaceho algoritmu. Problém ale ostáva, ak sa zmeny sústredia do jednej oblasti. Otázne ale ostáva aj to, či tento model zachovania mentálnej mapy hrán bude postačujúci aj v prípade veľkých - rozsiahlych zmien v grafe.

Kapitola 3

Model koridoru

V tomto modeli sa budeme snažiť prísť na prirodzený spôsob, ako obmedziť možnú zmenu cesty. Intuitívne by bolo potrebné vytvoriť oblasti pre každý cestu v grafe, v ktorých by boli povolené akékoľvek zmeny vo vykreslení. Na obrázku *3.1 - A* je vykreslená cesta spolu s návrhom takejto oblasti. Na obrázku *3.1 - B* jeden vrchol zmenil svoju polohu a došlo k prekresleniu. Zelená je cesta pred interakciou. Fialová je cesta, ktorú navrhuje kresliaci algoritmus. Čierna cesta je výsledná - po interakcií ostala v pomyslenej modrej oblasti a je zjavne viacej podobná ceste pred interakciou. Nasledujúca kapitola sa venuje problému, ako takúto oblasť pre ľubovolnú cest nájsť.



Obr. 3.1: *Obrázok A (vľavo) a obrázok B (vpravo)*

Hranu, ako sme si definovali, tvorí niekoľko bodov (0 až viac), v ktorých cesta mení smer. Pomocou týchto bodov môžeme rozdeliť cestu na niekoľko častí – niekoľko úsečiek – a pre každú takúto úsečku sa vytvorí **podkoridor**.

$$XY_P = \{V_0, V_1, V_2, ..., V_n\}$$

kde V_0 je vrchol X (kde cesta začína) a V_1 je vrchol Y (kde cesta končí)

Definícia 3. *Podkoridor pre úsečku V_iV_{i+1} (medzi bodmi V_i a V_{i+1}) tvorí usporiadaná štvorica bodov A, B, C, D. Tieto body vytvárajú štvoruholník, ktorý nazývame podkoridor.*

$$V_iV_{i+1}^{SC} = \{A, B, C, D\}$$

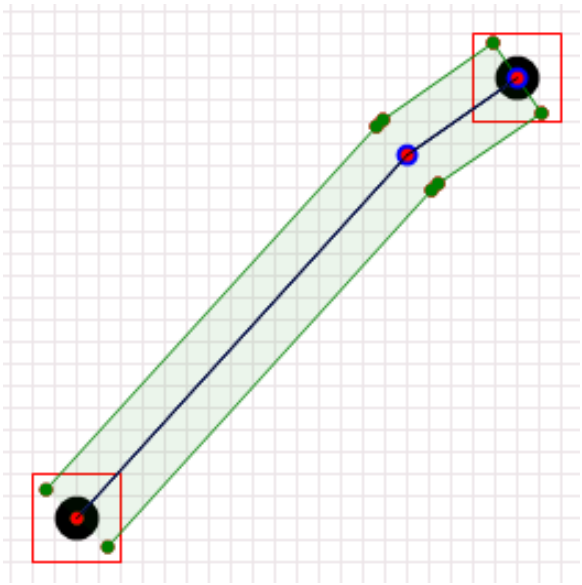
Pre body A, B, C, D platia nasledujúce podmienky :

- *Body A a B ležia na priamke p_1. Priamka p_1 prechádza bodom V_i a je kolmá na úsečku V_iV_{i+1}.*
- *Body C a D ležia na priamke p_2. Priamka p_2 prechádza bodom V_{i+1} a je kolmá na úsečku V_iV_{i+1}.*
- *Body A a B sú od bodu V_i vzdialené o rovnakú vzdialenosť (hodnota je konštantná pre všetky podkoridory). O rovnakú vzdialenosť sú vzdialené body C a D od bodu V_{i+1}*

Body A, B, C, D pre koridor označujeme aj ako **body charakterizujúce podkoridor**.

Pomocou týchto podkoridorov sa vytvára hľadaný **koridor**.

Po spojení týchto podkoridorov do jedného koridoru (mnohouholníka) vzniká oblasť, v ktorej dovolíme posun cesty. Mimo túto oblasť sa cesta dostať nesmie. Pri spájaní sa niektoré body charakterizujúce podkoridor môžu odstrániť (budú napr. obsiahnuté v inom podkoridore a preto budú nepotrebné). Vzniknú tak nové body, ktoré sa budú označovať podobne ako pri podkoridoroch - **body charakterizujúce koridor**. Spájať jednotlivé podkoridory by sa mohlo realizovať intuitívne. Výsledný



Obr. 3.2: *Príklad cesty, ktorá je rozdelená na 2 podkoridory. Čiernou farbou sú zvýraznené dva body, medzi ktorými sa hľadá cesta. Cesta mení smer v jednom bode – modro červený vrchol. Vznikajú teda dve úsečky a teda aj dva podkoridory. Podkoridory sú vyznačené zelene vyfarbenou oblasťou. Body pod-koridorov sú zvýraznené zelenou farbou.*

koridor sa začne budovať v smere z jedného koncového bodu cesty k druhému koncu cesty. Podkoridory úsečiek sa budú spájať vždy po dvoch – v poradí ako prechádzame úsečky. Koridor budú tvoriť vždy koncové body podkoridorov (teda výsledný koridor tvoria všetky body C a D z podkoridorov).

Definícia 4. *Množina podkoridorov (XY_P^{SC}) pre cestu je množina všetkých podkoridorov, ktoré sa pre cestu vytvorili.*

$$XY_P^{SC} = \cup V_i V_{i+1}^{SC} \ , \ kde \ 0 \leq i \leq n-1 \ a \ V_i, V_{i+1} \epsilon XY_P$$

Z všetkých bodov týchto podkoridorov je vytvorená množina bodov XY_P^{ALL}.

Definícia 5. *Koridor XY_P^{COR} vzniká spojením podkoridorov do jedného mnohouholníku (oblasti). Body tohto mnohouholníka sú podmnožinou množiny XY_P^{ALL}.*

$$XY_P^{COR} \subseteq XY_P^{ALL}$$

Tento postup nemusí byť vždy jednoznačný. Pri veľkých zmenách na ceste (napr.

cesta zmení smer o 90 stupňov a pod.) sa môžu body rôznym spôsobom prekrývať a spájanie podkoridorov do koridorov sa mení na zložitú geometrickú úlohu. Navyše, v ďalších častiach práce, bude veľmi užitočné vedieť koridor „škálovať“ pre rôzne parametre získať koridory rôznych vlastností (niekedy môžeme potrebovať koridor, ktorý bude totožný s konvexným obalom, inokedy bude problém koridor nájsť a pod.).

Pri voľbe vhodnej počiatočnej veľkosti podkoridoru – ktorú si môže zvoliť používateľ – a spojení do výsledného koridoru získame spôsob, ako efektívne nájsť cestu ktorá bude pri zachovávaní mentálnej mapy hrán spĺňať nasledovné podmienky :

- Cesta bude estetická – nebude prechádzať cez vrcholy, nebude sa križovať a pod. (závisí od vlastností kresliaceho algoritmu)
- Cesta sa zmení len o kontrolovanú vzdialenosť v každom svojom bode (zabezpečí nájdenie koridoru)

Splnenie týchto podmienok nám zabezpečí výsledok, ktorý by sme chceli dosiahnuť.

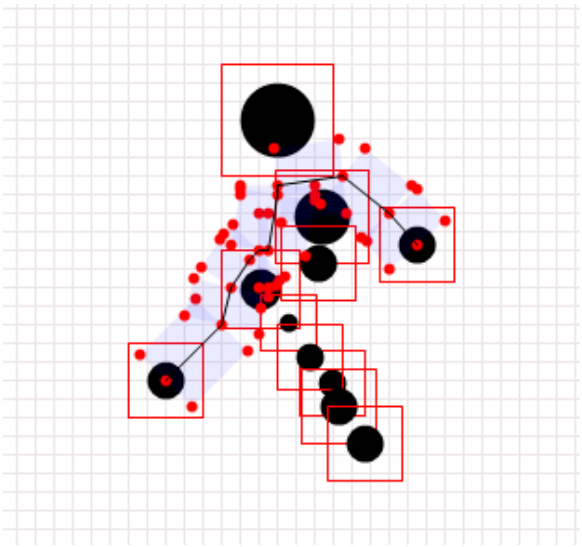
Tvrdenie 1. *Majme množinu bodov (a dané ich súradnice na x-ovej a y-ovej osi), body charakterizujúce podkoridory. Mnohouholník – polygón, ktorý bude obsahovať všetky dané bode, bude škálovateľný (veľkosť sa dá určovať pomocou vstupných koridorov) a hranami verne popisuje body z vstupnej množiny je koridor.*

Nasledujúce časti popisujú spôsob, ako takýto koridor nájsť.

3.1 Konvexný obal

Prvé a intuitívne riešenie ako vytvárať koridor sa ponúka nájdene konvexného obalu týchto bodov. Nájdene konvexného obalu je známy problém a na jeho riešenie bolo nájdených viacerých algoritmov. Pre posúdenie kvality koridoru získaného pomocou takého prístupu bol použitý takzvaný "Gift Wrapping“ algoritmus (implementované do pripravenej aplikácie).

Gift Wrapping algoritmus (známy aj ako **Jarvi‘s march**[13] algoritmus) bol nájdený v roku 1973. Jeho časová zložitosť je $O(nh)$, kde



Obr. 3.3: *Obrázok ilustruje situáciu, kedy jednoduché spájanie podkoridorov môže byť problém*

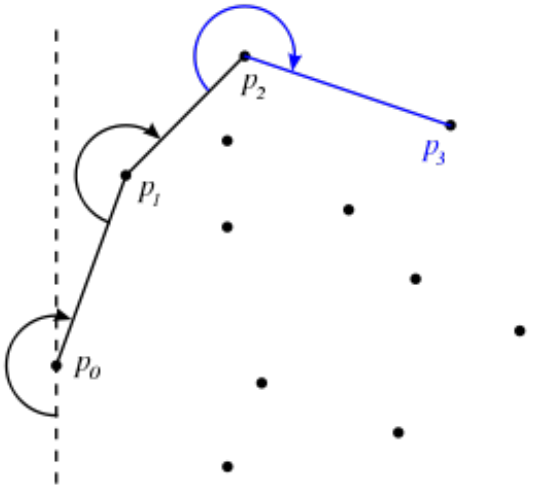
- n je počet bodov spomedzi ktorých hľadáme konvexný obal
- h je počet bodov, ktoré sa nachádzajú v konvexnom obale

V prvom kroku sa vyberie jeden bod (napr. najľavejší – ten s najmenšou x súradnicou) a druhý bod k nemu tak, aby všetky ostatné body ležali napravo od priamky, ktorú tieto body vytvárajú (príklad obrázok *3.4*). Tieto dva body vytvárať jednu hranu v konvexnom obale.

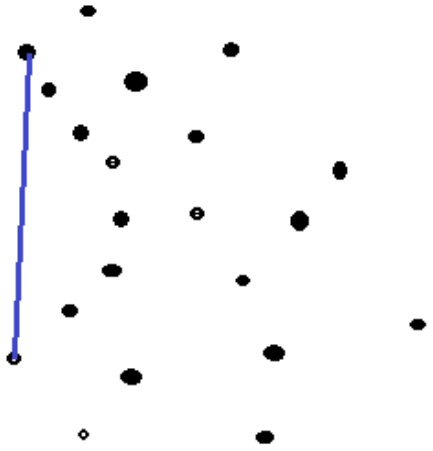
V ďalších iteráciách sa bude hľadať vždy nový bod, ktorý sa pridá do konvexného obalu, nasledovne : pre všetky body, ktoré sa nenachádzajú už v konvexnom obale sa vypočíta rozhodovací uhol. Uhol vytvárajú posledné dva body, ktoré boli pridané do obalu a jeden nový bod – kandidát. Počíta sa vonkajší uhol a vyberá sa ten vrchol, ktorý vytvára najmenší uhol (príklad obrázok *3.4*).

Nájdenie konvexného obalu ľahko rieši problém nájdenia koridoru. Ľahko ale vieme vytvoriť príklad, kedy nájdenie konvexného obalu nebude postačujúce. Navyše, konvexný obal pre danú množinu bodov je vždy len jeden a nedá sa pomocou parametrov ovplyvňovať.

Na obrázku *3.5* je vidno niekoľko bodov, pre ktoré keď nájdeme konvexný obal,



Obr. 3.4: *Jarvi's march algoritmus - jeho postup pri "balení"bodov*

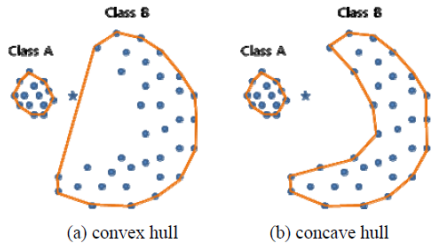


Obr. 3.5

bude obsahovať aj „modrú hranu“. Ako vidieť z obrázku, koridor by obsahoval aj veľkú oblasť navyše, kde sa žiadne body nenachádzajú. Koridor sa v tejto oblasti stane príliš "nafúknutý", čo môže mať za následok nesprávne zachovávanie mentálnej mapy hrán.

3.2 Upravený algoritmus hľadania konvexného obalu

Pre zlepšenie výsledkov je potrebné nájsť lepší spôsob ako vytvárať koridory a pridať možnosť určovať parametrov pre ovplyvňovanie hľadania koridoru. Ďalší použitý algoritmus bol od **JIN-SEO PARK** a **SE-JONG OH**[14]. Ich algoritmus vytváral n-dimenzionálne „konkávne obaly“.



Obr. 3.6: *Rozdiel medzi konvexným a konkávnym obalom*

Algoritmus vychádza z vypočítaného konvexného obalu s ktorým ďalej pracuje (upravuje ho) a parametra (označovaný ako N alebo T – *threshold*). Následne sa v niekoľkých iteráciách opakuje tkz. *„digging“* krok : Vyberie sa bod, ktorý leží vo vnútri konvexného obalu a od ostatných vnútorných bodov sa líši tým, že leží najbližšie k niektorej hrane konvexného obalu (má najmenšiu vzdialenosť od ľubovolnej hrany z konvexného obalu) – táto hrana sa zapamätá. Pomocou jednoduchého vzťahu medzi vzdialenosťami nového bodu a bodmi hrany sa algoritmus rozhodne, či dôjde k *„dig“* procesu. **Dig proces** – proces kedy je hrana roztrhnutá. Nech hranu tvorili dva body $P1$ a $P2$ a nový bod, na ktorom robíme dig operáciu je X. Odstránime hranu $P1P2$ a pridáme dve hrany – $P1X$ a $XP2$.

Pre tento algoritmus je veľmi dôležitá hranica N (threshold). Táto hranica ovplyvňuje, kedy bude hrana bodom X roztrhnutá a kedy nie.

Použitím tohto algoritmu sa nedosiahlo žiadneho výrazného zlepšenia oproti obyčajnému hľadaniu konvexného obalu. Zlepšením ale je, že algoritmus obsahuje parameter N, ktorým sa dá ovplyvňovať nájdený ”konkávny obal”.

3.3 Obal určený alfa hranami

Koridor budeme hľadať pomocou alfa hrán, ktoré ponúkajú veľmi dobre škálovateľné riešenie pre nájdenie koridoru charakterizujúci body v priestore.

Definícia 6. *Na vstupe pri hľadaní alfa kružníc je množina bodov a špeciálna alfa hodnota. Alfa hrana je úsečka medzi dvoma bodmi z tejto množiny, pričom vieme nájsť takú kružnicu*[1] *(alfa kružnicu), aby oba body ležali na kružnici a žiadny iný bod z vstupnej množiny bodov neležal ani na kružnici ani vo vnútri kružnici.*

Ak nájdeme alfa hrany na množine bodov, pomocou jednoduchších operácií (odstránenie „krížových hrán“, nájdene správneho koridoru a pod. - viac v nasledujúcej kapitole) získame škálovateľný spôsob ako hľadať koridory.

Pred samotným algoritmom na hľadanie alfa hrán uvedieme definície dvoch pojmov, ktoré úzko súvisia s týmto algoritmom.

3.3.1 Delaunayová triangulácia

Definícia 7. *Je to špeciálna triangulácia roviny[15] – teda rozdelenie roviny na trojuholníky – pre množinu bodov, spĺňajúce nasledujúce podmienky :*

- *vrcholy trojuholníkov tvoria vrcholy z vstupnej množiny*
- *ani jeden z množiny bodov neleží v kružnici opísanej ľubovolnému trojuholníku triangulácie*
- *triangulácia maximalizuje najmenšie uhly, ktoré vytvárajú trojuholníky*

Výsledkom sú teda trojuholníky, v ktorých neležia žiadne body z vstupnej množiny bodov. Pre body ležiace na priamke neexistuje žiadna triangulácia. Pre štyri a viac bodov ležiace na rovnakom kruhu (napr. vrcholy obdĺžnika) nie je Delaunayová triangulácia unikátna (vieme nájsť viac triangulácií roviny, ktoré spĺňajú podmienky).

[1] Pozn. : Veľkosť alfa kružníc priamo ovplyvňuje alfa hrany, ktoré budú nájdene

Veľmi jednoduchý algoritmus, ako nájsť takúto trianguláciu je vytvoriť ľubovolnú počiatočnú trianguláciu a následne v iteráciách hľadať chyby (porušenia podmienok) a následne upravovať trianguláciu (**"Flip algoritmus"**) - meniť strany / trojuholníky.

V algoritme potrebujeme overovať, či sa bod nachádza vo vnútri kružnici. To, či niektorý bod leží vo vnútri kružnici opísanej trojuholníku sa dá efektívne zistiť (ale iba v rovine / 2 rozmernom priestore) vypočítaním determinantu :

$$\begin{vmatrix} A_x & A_y, & A_x^2 + A_y^2, & 1 \\ B_x & B_y, & B_x^2 + B_y^2, & 1 \\ C_x & C_y, & C_x^2 + C_y^2, & 1 \\ D_x & D_y, & D_x^2 + D_y^2, & 1 \end{vmatrix} = \begin{vmatrix} A_x - D_x, & A_y - D_y, & (A_x^2 - D_x^2) + (A_y^2 - D_y^2) \\ B_x - D_x, & B_y - D_y, & (B_x^2 - D_x^2) + (B_y^2 - D_y^2) \\ C_x - D_x, & C_y - D_y, & (C_x^2 - D_x^2) + (C_y^2 - D_y^2) \end{vmatrix} > 0$$

Obr. 3.7: *Výpočtom tohto determinantu sa zistí, či bod leží vo vnútri kružnici opísanej trojuholníku*

Body A, B, C sú vrcholy trojuholníku zoradené v protismere točenia hodinových ručičiek a bod D je bod, ktorý overujeme, či leží v kružnici. Ak je determinant kladný, bod leží v kružnici.

Asymptotický algoritmus potrebuje $O(n^2)$ hranových zmien, aby sme získali správnu Delaunayovú trianguláciu (kde n je počet vrcholov).

Ak by sme spravili zjednotenie všetkých možných trojuholníkov, získame konvexný obal bodov. Delaunayová triangulácia sa dá použiť aj v n-rozmernom priestore a teda zjednotenie všetkých "(n) simplexov"získame (n+1)-rozmerný konvexný obal).

Delaunayovu trianguláciu budeme hľadať pre množinu bodov charakterizujúce podkoridory.

3.3.2 Voronoiove diagramy

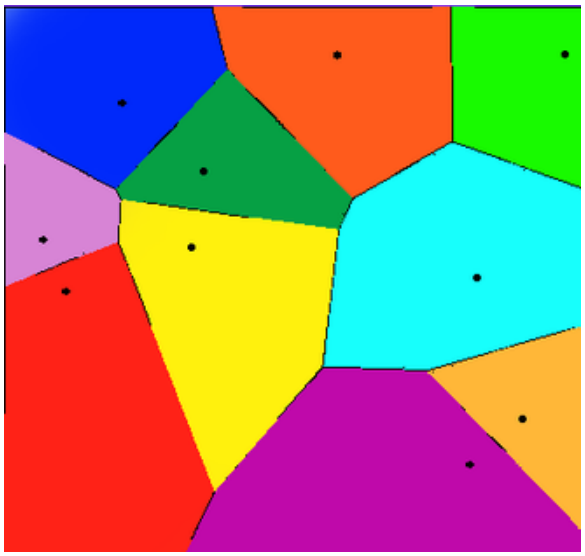
Jedná sa o špecifické rozdelenie priestoru do niekoľkých nových podpriestorov – oblastí (niekedy označované aj ako dekompozícia priestoru)[16].

Definícia 8. *Majme priestor X a funkciu d, ktorá dokáže medzi dvoma bodmi v tomto priestore zmerať vzdialenosť medzi nimi. Hľadáme množinu podpriestorov P (označujeme ich P_k). Pre každý tento podpriestor bude vyčlenený špeciálny bod A_k*

– môžeme ho volať „stred Voronoiovho pod-priestoru“, ktorý sa nachádza vo vnútri podpriestoru P_k. Body A_1 až A_n sú body, vzhľadom na ktoré počítame Voronoiove diagrami (sú dané na vstupe).

Pre každý podpriestor platí (nech je to k-ty podpriestor, čiže P_k podpriestor s „stredom“ A_k), že ľubovolný bod, ktorý leží v tomto podpriestore leží bližšie k A_k ako k inému A_j z podpriestoru P_j, pričom j je rôzne od k. Ak bod X :

- *leží v podpriestore A_k, potom $\forall j, j \neq k, d(X, P_k) < d(X, P_j)$*
- *ináč, $\exists j, j \neq k, d(X, P_j) < d(X, P_k)$*



Obr. 3.8: *Príklad vytvorených voronoiových diagramov*

Pre určenie vzdialenosť medzi dvoma bodmi používame vzorec pre euklidovskú vzdialenosť : $d(A, B) \equiv d(x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ (vzdialenosť medzi bodom $A[x_1, y_1]$ a bodom $B[x_2, y_2]$).

Pomocou Voronoiových diagramov vieme získať konvexný obal. Dva body z vstupnej množiny budú v konvexnom obale spojené práve vtedy ak :

- ak Voronoiove podpriestory, do ktorých tieto body patria, sú susedia (majú aspoň jednu hranu spoločnú)

- ich spoločná hrana je nekonečne dlhá polpriamka

Pre body charakterizujúce podkoridory budeme hľadať Voronoiove diagramy. V nasledujúcej časti sa ukáže, ako hľadanie týchto diagramov súvisí s hľadaním Delaunayovej triangulácie a ako to pomôže pri hľadaní koridoru.

3.3.3 Hľadanie Voronoiových diagramov pomocou Delaunayovej triangulácie

Algoritmus, ako nájsť Delaunayovu trianguláciu sme už uviedli. Teraz ukážeme, ako pomocou tejto triangulácie nájdeme Voronoiove diagramy.

Voronoiove diagramy vytvoria z priestoru niekoľko nových podpriestorov P_k, ktoré sú určené svojim "Voronoyov-im stredom" A_k. Je potrebné určiť :

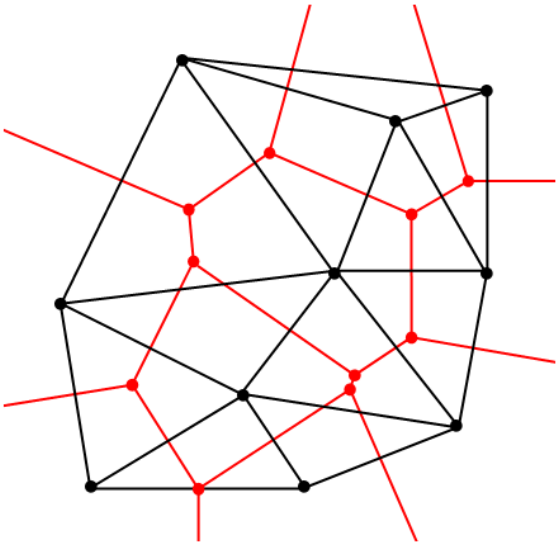
- podpriestory (voronoiove diagrami) a ich stredy
- ktoré podpriestory sú susedné

Stredy Voronoiových diagramov sú stredy opísaných kružníc trojuholníkov, ktoré vznikli pri triangulácii. Problém susednosti podpriestorov vyriešime nasledovne. Dva podpriestory P_k (s stredom A_k) a P_j (s stredom A_j) sú susedné, ak trojuholníky, ktorých stred opísanej kružnice bol bod A_k alebo A_j, mali spoločnú hranu.

Problém susednosti sa 'do istej miery' medzi Voronoiovými diagramami a Delaunayovu trianguláciu zachováva. Ak boli dva trojuholníky pri triangulácii susedné (mali spoločnú jednu hranu), susedné budu aj Voronoiové diagramy, ktoré vzniknú pomocou týchto trojuholníkov.

3.3.4 Alfa hrany

Nájdenú trianguláciu a rozdelenie priestoru na podpriestory použijeme pri hľadaní alfa hrán[17] a alfa kružníc. Alfa kružnica je charakterizovaná polomerom – alfa hodnotou. Alfa kružnica musí obsahovať na obvode práve dva body, ktoré sú spojené v



Obr. 3.9: *Obrázok ilustruje problém susednosti. Čiernou farbou sú nakreslené trojuholníky z triangulácie. Červenou farbou sú nakreslené hrany Voronoiových diagramov. Červené bodky sú stredy opísaných kružníc trojuholníkov. Spojené (červenou) sú vtedy, ak sú trojuholníky susedné.*

triangulácií (teda niektoré dva body A_k a A_j) a žiaden ďalší iný bod (ani na kružnici ani v kružnici).

Ak medzi dvoma bodmi, ktoré sú spojené triangulációu existuje alfa kružnica (resp. ju vieme zostrojiť), hrana bude aj v polygóne – koridore. To, ako bude vyzerať takýto polygón oplyvňuje alfa parameter – polomer kružníc (keďže nie vždy sa dané kružnice dajú nájsť).

Pre úplnosť : parameter alfa (α) môže byť

- $\alpha < 0$, vtedy je polomer kružnice $-1/\alpha^2$ a koridor je hľadaný popísaným spôsobom
- $\alpha = 0$, vtedy je to konvexný obal
- $\alpha > 0$, vtedy je to polygón s väčším obsahom, ako konvexný obal – konštrukcia takéhoto polygónu je takmer identická ako pre $\alpha < 0$ (mierne upravené sú

[2]Polomer bude kladné číslo

hľadania triangulácie a Voronoiových podpriestorov). Polomer alfa kružníc je $1/\alpha$.

3.4 Hľadanie koridorov pomocou alfa hrán

Samotné aplikovanie algoritmu na hľadanie alfa hrán nie je postačujúce pre nájdenie správneho koridoru. Problémom sú nadbytočné hrany, ktoré je potrebné odstrániť. Nasledujúci algoritmus popisuje dôležité problémy, ktoré treba vyriešiť, aby sa koridor dal úspešne nájsť :

Algoritmus pre nájdenie koridoru

(Vstup je cesta - množina bodov)

1. *Rozšírenie podkoridorov*
2. *Vypočítanie alfa hrán pre body podkoridorov*
3. *Spracovanie alfa hrán*
    - *Určenie alfa hodnoty a získanie alfa hrán*
    - *Transformácia alfa hrán na graf*
    - *Rozdelenie grafu na tri komponenty*
    - *Prehľadávanie komponent grafov*
4. *Vytvorenie koridoru alebo nastavenie novej alfa hodnoty (znova k bodu 3.)*

(Výstupom je množina bodov, ktoré vytvárajú koridor)

V nasledujúcej časti sú jednotlivé kroky bližšie popísane spolu s navrhovaným postupom ako vyriešiť problémy, ktoré z nich vyplývajú.

3.5 Rozširovanie podkoridorov

Pri vytvorení základných podkoridorov sa môže stať, že niektoré vrcholy grafu budú zasahovať do práve vytvoreného (alebo aj viacerých) podkoridorov. V takom prípade je ale podkoridor obmedzený – keďže cez daný vrchol sa cesta nemôže viesť, prekážka sa musí obísť.

Ďalším krokom bude **rozšíriť podkoridory** tak, aby prekážka bola celá obsiahnutá v podkoridore. V takom prípade pri hľadaní cesty môžeme prekážku obísť. Ak sa po rozšírení narazí na novú prekážku, podkoridor sa musí znova rozšíriť. Ak by ale nová prekážka bola už celá obsiahnutá v podkoridore, rozširovanie sa nemusí nutne udiať. Pri tejto kontrole sa môžu brať do úvahy aj iné podkoridory (vo výsledku sa ich plochy zjednotia). Preto, ak by časť prekážky nebola v podkoridore obsiahnutá, ale v inom bola, môžeme prekážku považovať za obsiahnutú – rozširovanie sa nemusí udiať.

Kontrola prekážky

V implementácií práce sú vrcholy reprezentované ako kruhy. Keďže vrcholy môžu mat rôzny tvar (trojuholníky, 5-uholníky ...) rôznej veľkosti, vrcholy obklopuje štvorcový rámec – tento štvorec zjednodušuje prácu pri riešení úloh ako :

- zistenie, či bod leží vo vrchole
- vypočítanie priesečníka úsečky a vrchola

Overenie, či vrchol leží v podkoridore je teda overenie, či všetky jeho úsečky štvorcového rámca ležia v podkoridore.

Definícia 9. *Kontrola prekážky je overovanie vplyvu vrcholu A na všetky podkoridory v množine XY_P^{SC}, pričom vrchol A má štvorcový rámec tvorený bodmi $A = A_1, A_2, A_3, A_4$. Podkoridor V_iV_{i+1} sa nemusí rozširovať (vzhľadom na vrchol A) ak každá úsečka rámca (A_1A_2, A_2A_3, A_3A_4, A_4A_1) leží aspoň v jednom podkoridore XY_P^{SC} cesty XY_P.*
Kontrola sa vykonáva vzhľadom na všetky vrcholy v grafe.

Ak by sa namiesto štvorcových rámcov zvolila iná reprezentácia, bolo by potrebné upraviť túto kontrolu. Jej myšlienka by sa ale zachovala. Taktiež, ak by kontrola mala byť skutočne dôsledná, mohla by byť navrhnutá kontrola každého bodu na úsečke (body s súradnicami z prirodzených čísel – aby počet bodov bol konečný). Každý takýto bod by musel ležať v niektorom podkoridore.

Po rozšírení podkoridorov treba ešte vyriešiť niekoľko nasledujúcich problémov, kým sa dosiahne hľadaný koridor. Tieto problémy sú :

- Určovanie správnej alfa hodnoty
- Z množiny alfa hrán vybrať hrany popisujúce koridor
- Overenie nájdeného koridoru

3.6 Vypočítanie alfa hrán pre body podkoridorov

To, ako pre body v priestore vypočítať útvar – polygón, ktorý bude verne opisovať (tzv. konkávny obal) body, bolo ukázané v časti 3.3.4. Pre všetky podkoridory sa vyberú body (body charakterizujúce podkoridory) a na tieto body sa aplikuje algoritmus hľadania alfa hrán.

3.7 Určovanie alfa hodnoty

Dva body medzi sebou majú alfa hranu práve vtedy, keď vieme nájsť vhodnú alfa kružnicu. Polomer alfa kružnice určuje zvolenie alfa hodnoty. Vzniká preto problém, ako správne zvoliť túto hodnotu, aby sme vedeli skonštruovať hľadaný koridor. Ako jednoduché riešenie by sme mohli použiť algoritmus, ktorý by iterovaným princípom stále znižoval alfa hodnotu, kým by nenašiel nami hľadaný výsledný polygón – koridor. Týmto spôsobom sa zamedzí vzniku niekoľkých malých disjunktných koridorov. Naším cieľom je nájsť koridor, v ktorom sa budú nachádzať oba body, pre ktoré hľadáme cestu.

Ako počiatočná hodnota sa môže zvoliť hodnota, pri ktorej sa získajú veľmi malé kružnice (napr. – 1000, prehľadávať budeme iba v intervale záporných hodnôt z dôvodu vlastností algoritmu hľadania alfa hrán[3]). Postupne zvyšujeme hodnotu alfa (čím sa zväčšuje polomer alfa kružníc). V prípade hustých grafov (alebo klastrov/zoskupených vrcholov) nám tieto hodnoty budú postačovať na nájdenie polygónu.

Keďže sa jedná o iterovaný postup, je potrebné vedieť určiť, kedy sa má algoritmus zastaviť – určiť alfa hodnotu, kedy už nemá ďalej význam prehľadávať menšie a menšie hodnoty. Cesta medzi dvoma bodmi môže prechádzať v blízkosti skupiny vrcholov (kde v dôsledku zväčšovania sa koridoru vznikne viacej bodov), no samotné body (medzi ktorými hľadáme cestu) môžu byť vzdialené – alfa kružnice s malým polomer by nám vyhovovali v skupinke vrcholov, no vzdialené body by neboli spojené a teda by sme nevedeli vytvoriť rozumný výsledný koridor.

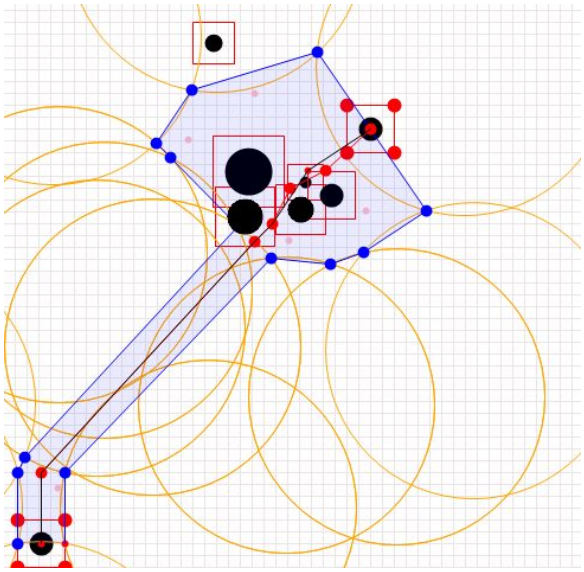
Z popísaného problému vidieť, že jeden z "najhorších"prípadov nastane, keby sme dva body, ktoré chceme spojiť cestou, umiestnili čo najďalej od seba (opačné rohy na obdĺžnikovej kresliacej ploche). Nech táto vzdialenosť je x. Veľkosť polomeru alfa kružnice asi nikdy nemusí byť väčšia ako táto hodnota x – kružnice sú už tak veľké, že budú vedieť „spájať“ aj tie najvzdialenejšie body. Keď si vytvoríme jednoduchý vzťah :

$$\frac{-1}{-alfa} = x$$

ľahko vypočítame alfa hodnotu, za ktorou už nemusíme ďalej prehľadávať. Pri takýchto hodnotách sa získavajú už viac menej konvexné obaly z vrcholov.

Nevýhodou takýchto veľkých alfa kružníc je, že skupiny vrcholov/klastre nemajú detailné koridory. Veľké kružnice majú tendenciu zobrať niektoré krajné vrcholy koridorov častí ciest a tie spojiť - z týchto bodov sa vytvára často konvexný obal.

[3]Pozn. : už pri alfa hodnote -1000 máme kružnice s polomer -1 / - 1000, teda 0.001, čo sú veľmi malé kružnice vhodné pre klastre vrcholov



Obr. 3.10: *Obrázok ilustruje príklad, kedy malé alfa hodnoty nemusia byť dostatočné, aby vznikli alfa kružnice, ktoré by nám vytvorili správny hľadaný koridor. Na obrázku už bola aplikovaná metóda iterácie, aby sa našla správna alfa hodnota.*

Iné riešenie hľadania alfa hodnôt

Keďže nevieme nájsť pomocou veľkých alfa kružníc detailné koridory v klastroch vrcholov a zas malé kružnice detailne popisujú zoskupenia skupín vrcholov nevedia spojiť vzdialené hrany a vytvoriť tak vhodný výsledný polygón ponúka sa jednoduchý navrhovaný postup[4].

Stačí keď rozdelíme problém hľadania koridoru na niekoľko pod problémov dvoch typov :

- skupiny vrcholov – klastre
- vzdialené vrcholy / priestor medzi vzdialenými klastrami

Každý jeden vzniknutý pod problém riešime samostatne (pri klastroch sa vyberie

[4]Javí sa veľmi užitočné ponúknuť používateľovi možnosť nastavenia najmenšieho možného polomeru kružnice (od tohto polomeru sa ďalej prehľadávajú možnosti) – resp. používateľovi sa ponúkne možnosť nastavovanie „kvality detailov“ koridorov, aby používateľ mohol abstrahovať od znalostí hľadania alfa kružníc.

menšia alfa hodnotu, pri vzdialených väčšia hodnota).

3.8 Spracovanie alfa hrán

Výsledok (množina alfa hrán), ktorý je na výstupe predchádzajúceho kroku má potenciálne dve chyby, ktoré musíme vedieť odstrániť :

- Nepodarilo sa dostatočne odstrániť vznik veľkého počtu malých disjunktných koridorov
- Z alfa hrán sa nemusí dať vytvoriť koridor (obsahuje príliš veľa zbytočných hrán)

V ďalšej časti sa ukáže spôsob, ako dané problém vyriešiť.

3.8.1 Transformácia na graf

Množina alfa hrán sa intuitívne transformuje na graf. Vrcholy podkoridorov budú vrcholy v grafe. Hrana medzi dvoma vrcholmi existuje práve vtedy, ak medzi nimi existovala alfa hrana.

3.8.2 Rozdelenie grafu na tri komponenty

Graf, ktorý vznikol po transformácií, sa následne rozdelí na tri komponenty. Najprv sa ale vytvoria dve množiny vrcholov – množina ľavých a množina pravých vrcholov, ktoré budú tvoriť dva komponenty. Všetky ostatné vrcholy budú v tretej komponente grafu.

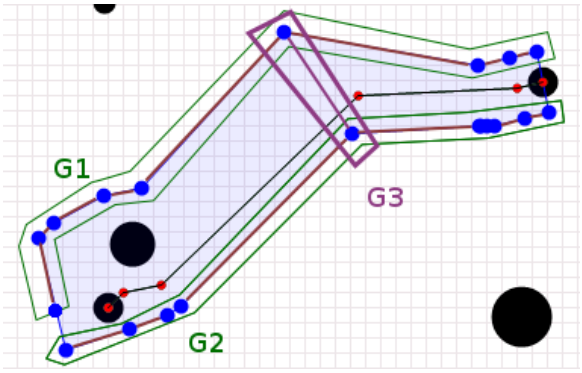
Cieľom bude rozdeliť vrcholy tak, aby v množine ľavých vrcholov boli iba tie vrcholy, ktoré, sa vždy nachádzajú „naľavo“ od cesty (komponenta G1) – keď sa cesta sleduje z bodu X do bodu Y. Presne rovnako sa vytvorí množina pravých vrcholov (komponenta G2).

Ak sa niekde cesta pretínala – vytvorila akúsi slučku[5] – v jej strede sa môžu vyskytnúť vrcholy medzi ktorými môžu existovať hrany. Tieto vrcholy budú zaradené do tretej komponenty (G3). Do tretej komponenty sa ešte priradia ďalšie vrcholy po dvojiciach (vrcholy A a B) ak :

- vrchol A je z ľavej množiny a vrchol B je pravej množiny (prípadne opačne)
- medzi vrcholmi A a B existuje hrana (teda tie vrcholy, ktoré tvoria prepojenia medzi komponentami G1 a G2)

Vrcholy komponenty G1 si zachovajú hrany medzi vrcholmi (navzájom) a taktiež aj hrany v G2 (iné hrany nie sú v komponentoch). V komponente G3 sú „mostové“ hrany medzi G1 a G2 a hrany medzi vrcholmi z množiny G3, ktoré nie sú v G1 a ani v G2 (teda ak cesta vytvorila slučky v ktorých sa nachádzali vrcholy, mohli tieto body v slučkách spájať hrany).

Všetky hrany z pôvodného grafu G sa rozdelili do troch komponent. Zjednotením týchto troch komponent by sa získal pôvodný graf G.



Obr. 3.11: *Ukážka rozdelenia alfa hrán do troch komponentov.*

[5]Bodov v slučkách veľmi veľa nikdy nebude. Keďže podkoridory sa najprv nafukujú, body v slučkách „preskočia“ mimo – von z slučky. Ak by vznikla veľká slučka, problém sa vyrieši zaradením vrcholov v slučke do komponenty G3.

3.8.3 Prehľadávanie komponenty grafu pre nájdenie koridoru

V komponentoch G1 a G2 sa bude hľadať ťah medzi dvoma vrcholmi (každý vrchol a hrana sa v ťahu môžu vyskytnúť iba raz). V komponente G1 bude jeden vrchol bod A z prvého podkoridoru a druhý vrchol bude bod B z posledného podkoridoru. Podobne pre komponentu G2. Ak sa takéto ťahy nájdu, našli sa dve "polovicu"z hľadaného koridou. Tieto dve polovice sa už len spojia a vytvoria hľadaný koridor.

Ak by sme v oboch komponentoch nenašli ťah, algoritmus sa musí vrátiť niekoľko krokov dozadu. Zmení alfa hodnotu – aby sa zväčšili veľkosť alfa kružníc. Vznikne nová množina alfa hrán, ktorá sa transformuje do grafu. Graf sa ďalej rozkladá do troch komponentov v ktorých sa hľadajú ťahy. Pri postupnom zväčšovaní alfa hodnoty sa kružnice čoskoro stanú tak veľké (ich polomer narastie), že alfa hrany budú vytvárať konvexný obal.

Ak by sa koridor stále nedarilo nájsť ani pri konvexnom obale, treba podkoridory viacej rozšíriť. Algoritmus sa vracia spätne k prvému bodu algoritmu. V najhoršom prípade sa podkoridory rozšíria natoľko, že sa v koridore obsiahne celá kresliaca plocha.

Kapitola 4

Hodnotenia a meracie techniky

Merať, alebo určovať kvalitu cesty pri zachovávaní mentálnej mapy hrán sa javí ako zložitý problém. Okrem subjektívneho pohľadu (každá osoba môže mať rôzne estetické požiadavky, schopnosť zachytiť zmeny vo vykreslenom grafe či skúsenosti pri práci s grafmi) sme preto navrhli niekoľko meracích techník. Tieto techniky porovnávajú rôzne geometrické vlastnosti ciest.

Priebežne, počas práce s vykresleným grafom, sme postupne budovali štatistiky. Pre každú meraciu techniky sme počítali priemer (aby sme zachytili jej celkový prínos pri zachovávaní mentálnej mapy) a jednotlive namerané hodnoty sa ukladali.

Porovnávanie a meranie sa konalo pri každej interakcii používateľ s grafom. Ak došlo k zmene vykresleného grafu, počítali sa nové hodnoty pre všetky ceste v grafe. Pri každom meraní sa pracovalo s troma cestami :

- pôvodnou cestou pred interakciou
- cestou od kresliaceho algoritmu (navrhovaná cesta)
- výslednou cestou, ktorá zachováva mentálu mapu cesty.

Predovšetkým nás zaujímala navrhovaná cesta a výsledná cesta - tieto cesty sme porovnávali a skúmali, ktorá je lepšia.

4.1 Podobnosť ciest

Toto meranie počítalo, ako sa jedna cesta podobá druhej ceste resp. koľko majú spoločných bodov.

Táto technika má na vstupe dve cesty (**meraná cesta** a **vzorová cesta**) a iteratívne vykonáva niekoľko jednoduchých meraní. Na meranej ceste sa vyberá vždy jeden bod. Na začiatku je to bod, ktorý sa nachádza v strede celej cesty – stredový bod (vzhľadom na celkovú dĺžku cesty). V ďalšej iterácii merania stredový bod rozdelí cestu na dve časti. Rekurzívne pokračujem ďalej s vzniknutými časťami (znova sa hľadá stredový bod). Hĺbku rekurzie určuje konfiguračný parameter. Celkový počet stredových bodov, ktoré takto vzniknú, je: $2^{paramater}$

Pre každý stredový bod sa overí, či leží niekde na vzorovej ceste. Pre každú úsečku vzorovej cesty sa vypočíta, či vzdialenosť medzi úsečkou a stredovým bodom je rovná 0. Ak áno, bod leží na ceste.

Na záver sa meranie vyjadrí v percentách :

$$\frac{\text{počet stredových bodov na vzorovej ceste}}{\text{celkový počet stredových bodov}} * 100$$

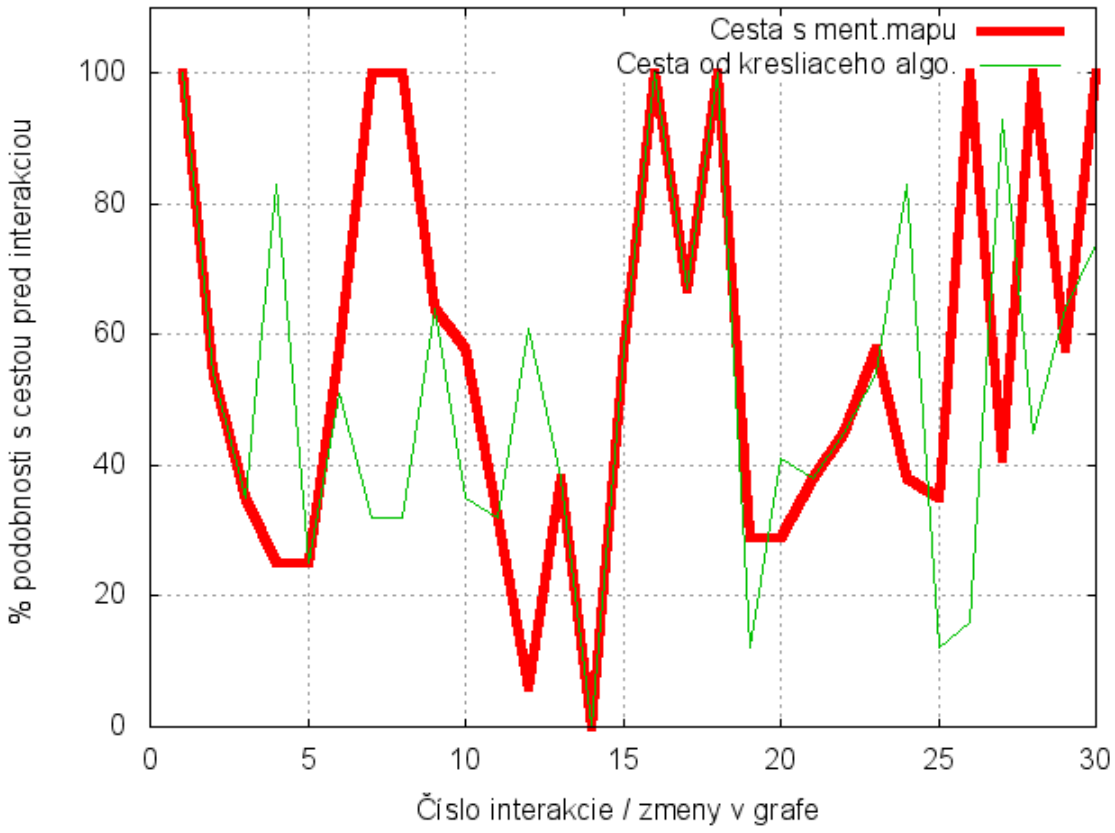
Pri interakcii s grafom sme pre každú cestu vykonali meranie :

- Navrhovanú cestu s pôvodnou cestou pred interakciou
- Výslednú cestu s pôvodnou cestou pred interakciou

Výsledok

Namerané hodnoty sa pomocou programu Gnuplot zobrazovali do grafov.

Na obrázku 4.1 sa dá spozorovať, ako sa namerané hodnoty pri zachovávaní mentálnej mapy hrán pomocou koridorov pohybovali mierne na vyšších hodnotách (červená krivka), zatiaľ čo bez mentálnej mapy boli bližšie k x-ovej osy (k nule).



Obr. 4.1

Čím je nameraná hodnota vyššia, tým je cesta viac podobná ceste pred interakciou. Použitie mentálnej mapy malo za následok zvýšenie nameraných hodnôt. Bez použitia mentálnej mapy bola podobnosť s cestou pred interakciou 51.93%. S použitím mentálnej mapy sa dosiahlo zvýšenie podobnosti na 56.66%, čo je približne o 5% viac.

Ak by sa zvolila iná interakcia s grafom, mohol by sa dosiahnuť lepší výsledok. Pri inej interakcií by zas zachovávanie mentálnej mapy nebolo potrebné a namerané hodnoty by boli identické. Všetky merania v práci sa konali pri identickej interakcií - v aplikácii sa nastavila sekvencia krokov, ktorá sa dá kedykoľvek spustiť.

4.2 Priemerná vzdialenosť

Cesta sa bude znova binárne rozdeľovať (na dva rovnaké diely). Pri každom delení budeme určovať vzdialenosť stredného bodu úseku cesty od pôvodnej cesty. Vzdialenosti sa spočítavajú a následne sa vydelia počtom delení cesty. Získame takto priemernú vzdialenosť cesty B od cesty A. Počet meraní znova ovplyvní parameter, ktorý určí ako hlboko sa budeme v rekurzií vnárať.

Ak by cesta B bola identická s cestou A, výsledkom bude 0 (prípadne hodnota veľmi blízka 0) – keďže každý stred ktoréhokoľvek úseku cesty B bude ležať na niektorej časti cesty. Cieľom tejto metódy je určiť, ako veľkú oblasť dve cesty medzi sebou uzatvárajú – odhadovanú vzdialenosť. Čím väčší počet meraní, tým presnejší bude výsledok.

Znova budeme pri každej interakcií s grafom počítať pre každú cestu v grafe dve merania :

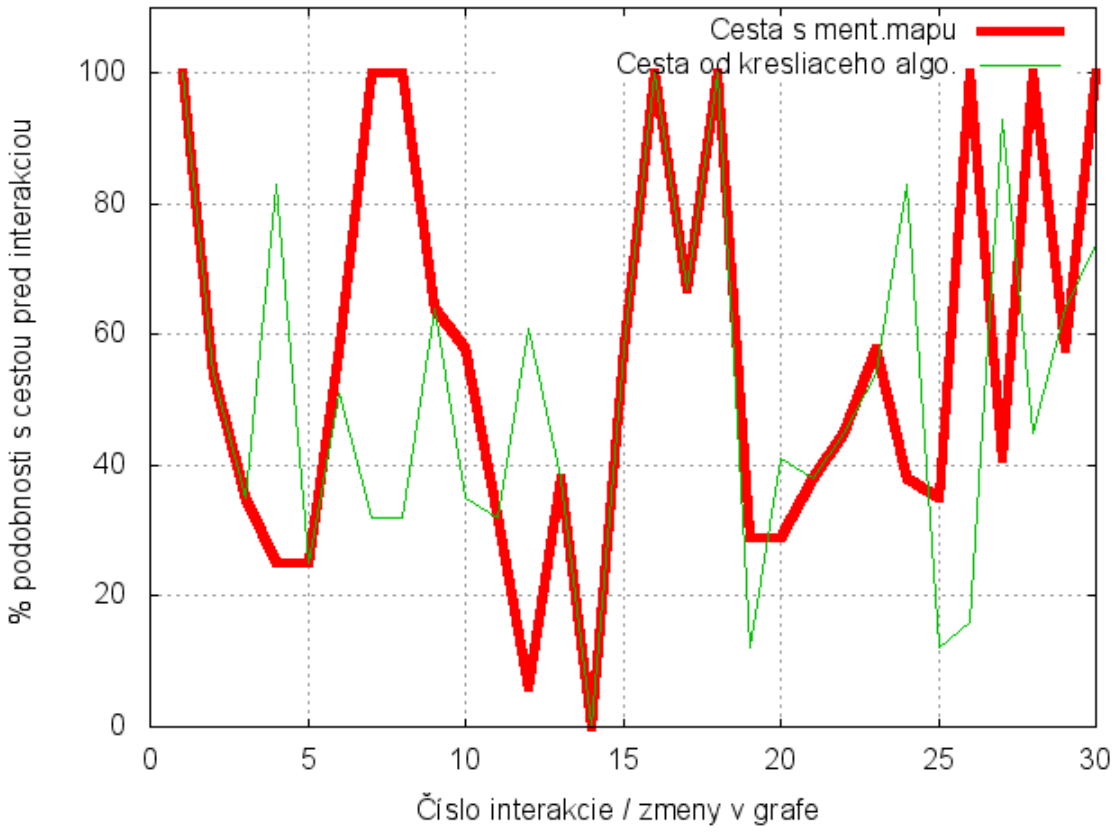
- Priemerná vzdialenosť navrhovanej cesty a cesty pred interakciou s grafom
- Priemerná vzdialenosť výslednej cesty od cesty pred interakciou s grafom

Výsledok

Namerané hodnoty sa znova premietli do grafu. Na obrázku 4.2 sa dá pozorovať, ako červená krivka (mentálnej mapa hrán sa zachováva znova pomocou koridorov) sa napr. len dva razy dostala pod úroveň 20 podobnosti s vzorovou cestou (nameraná priemerná vzdialenosť bola 17.07). Bez použitia mentálnej mapy to bolo viac krát (nameraná priemerná vzdialenosť bola 21.86).

4.3 Pomer dĺžok

Ak by sme skutočnú vzdialenosť cesty dali do pomeru s vzdialenosťou najkratšej možnej cesty (medzi dvoma bodmi je to úsečka), dostaneme číslo, ktoré považujeme,



Obr. 4.2

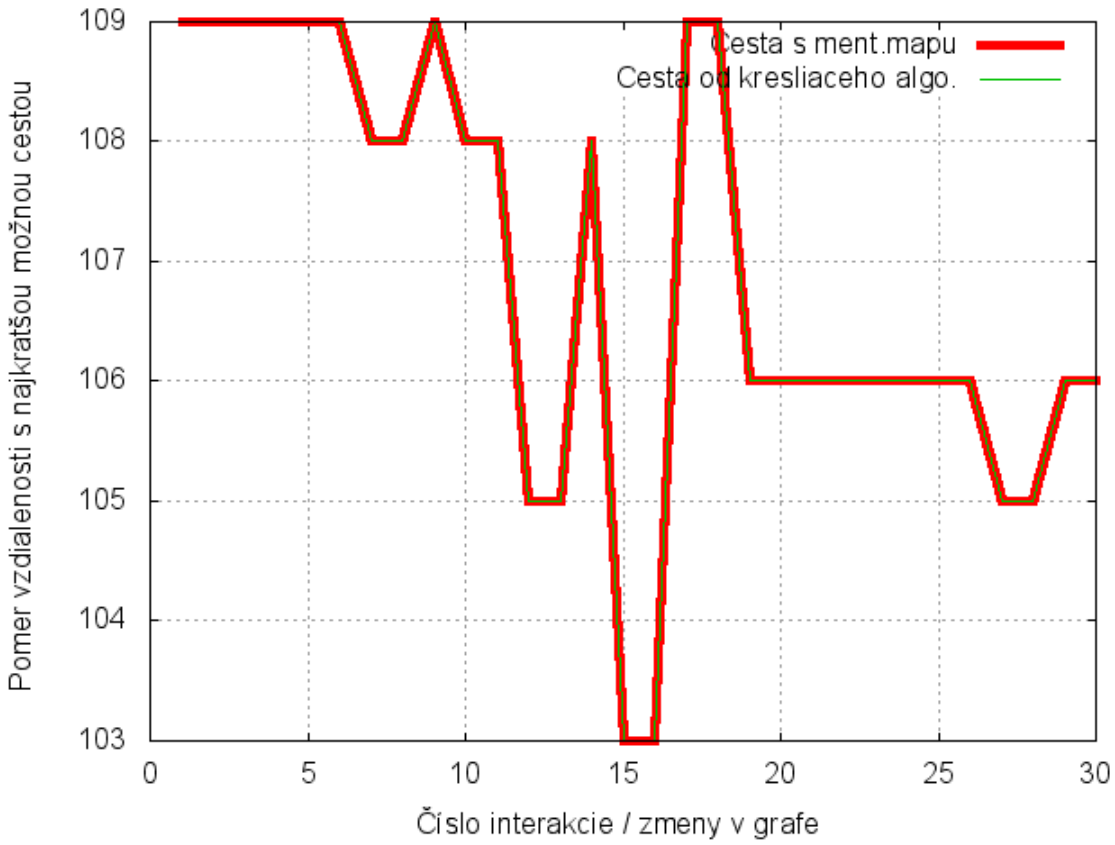
ako cesta bola veľmi efektívna. Samotné číslo samo veľa neznamená, no keď ho začneme porovnávať s výsledkom z predchádzajúcej interakcie, vieme vytvoriť meranie na základne jednoduchej hypotézy:

V ideálnom prípade by sme chceli, aby zložitosť cesty bola čo najmenšia – cesta má byť čo najjednoduchšia, najkratšia a pod. Túto hodnotu (pre hodnoty od výslednej cesty) by sme teda chceli minimalizovať (alebo aspoň nech v priemere nepresahuje výsledky cesty od kresliaceho algoritmu).

Podobne ako v predchádzajúcich technikách, merať sa bude :

- Pomer dĺžok navrhovanej cesty a cesty pred interakciou s grafom
- Pomer dĺžok výslednej cesty od cesty pred interakciou s grafom

Výsledok



Obr. 4.3

Na obrázku 4.3 vidno, že cesta so zachovávaním mentálnej mapy a bez zachovávania sú takmer identické. Namerané hodnoty boli 1.10 pri zachovávaní ment. mapy a 1.07 bez zachovávania. Cesta medzi dvoma vrcholmi by mala byť vždy jednoduchá, prehľadná a priamočiara. Pri zachovávaní mentálnej mapy sa cesta mierne predĺžila. Toto predĺženie objavilo ale až meranie a voľným okom sme nepostrehli žiadny veľký rozdiel (postrehli sme len cesty rozdielnych tvarov).

Prehľad výsledkov

Tabuľka 4.1: *Hodnoty, ktoré boli namerané pri zachovávaní mentálnej mapy s koridorom a bez zachovávania*

	Mentálna mapa (koridor)	Bez mentálnej mapy	Obmedz. počtu nových vrchol.)
Podobnosť ciest	56.66%	51.93%	71.46%
Priemerná vzdialenosť	17.07	21.86	10.99
Pomer dĺžok	1.1	1.07	1.2

Pri implementácií modelu zachovávania mentálnej mapy s obmedzovaním počtu novych vrcholv sa cesta medzi novou častou cesty a napojením sa na starú cestu (vrcholy $V_{a+K} \in XY_{Algo}$ a $V_{a+K+1} \in XY_P$) jednoducho spojila. Čo malo za následok dobré výsledky pri meraniach no mnohokrát viditeľné estetické nedostatky (napr. hrana často pretínala vrcholy).

Kapitola 5

Aplikácia

Testovanie a meranie oboch modelov sa robilo na aplikácii, ktorá bola pre tento účel vyvinutá. Program je napísaný v jazyku C# a vyvíjaný bol v programovacom nástroji Visual Studio 2010 - študentská verzia.

5.1 Popis aplikácie

Aplikácia je rozdelená na dve časti: vizualizačná plocha a panel nástrojov. Vizualizačná plocha ukazuje vykreslený graf. Na tejto ploche môže používateľ pracovať s grafom. Keď sa myš dostane nad vrchol, zobrazí sa jeho poloha /súradnice a identifikačné číslo. S vrcholmi môže používateľ hýbať, pridávať nové hrany či vrcholy alebo mazať hrany a vrcholy.

Panel nástrojov je rozdelený na 5 záložiek. Prvá záložka slúži na ovládanie interakcie s grafom. Dá sa tu nastaviť model zachovávania mentálnej mapy, upraviť štruktúru grafu, spustiť automatické scenária a exportovať namerané hodnoty do textových súborov. Pri exporte sa ukladajú všetky priebežné namerané hodnoty (pre 3 meracie techniky). Pre každú meraciu techniku sa ukladajú hodnoty pre cestu navrhovanú kresliacim algoritmom a pre výslednú cestu (na ktorú bola aplikovaná mentálna mapa).

Druhá záložka ovláda všeobecné nastavenia mentálnych máp. Zapína sa tu kreslenie ciest pred interakciou s grafom (označovane ciest pred interakciou s grafom) a navrhovaných ciest od kresliaceho algoritmu.

Na tretej a štvrtej záložke sa nastavujú parametre pre modely mentálnych máp.

Piata záložka ukazuje výsledky meraní, ktoré sa aktualizujú automaticky po každej interakcií s grafom. Pre každú z troch meracích techník sa dá pozrieť priemerná hodnota, ktorá sa získava od počiatku práce s aplikáciou. Prítomný je aj počet meraní a posledná nameraná hodnota.

5.2 Časti aplikácie

Na hľadanie ciest medzi dvoma bodmi sa použil mierne modifikovaný algoritmus A*. Tento algoritmus nájde medzi dvoma bodmi vždy tu najkratšiu cestu, ak cesta existuje. Pri hľadaní cesty sa algoritmus vyhýba vrcholom, resp. ich štvorcovým rámcom, ktoré vymedzujú plochu do ktorej nemôže cesta zasahovať. Po nájdení je cesta ďalej spracovaná (do formy cesty, postupnosť úsečiek : $XY_P = \{V_0, V_1, V_2, ..., V_n\}$). Výstupom a star algoritmus je sekvencia bodov (množina bodov), každý od seba vzdialený o konštantnú[1] vzdialenosť. Pri spracovaní sú body, ktoré ležia na rovnakej úsečke a vo výstupnej sekvencií sú za sebou, vymazané. Po tomto kroku teda cestu tvoria iba body, v ktorých cesta mení smer. Cesta sa dá vytvoriť jednoduchým pospájaním týchto bodov v správnom poradí.

Súčasťou aplikácie je aj interaktívna dokumentácia určená pre internetové prehliadače. Dokumentácia je automatický generovaná z komentárov v kóde aplikácie - ponúka teda prehľad všetkých použitých tried, metód, atribút a pod. (dokumentácia je určená pre prípadné ďalšie rozširovanie aplikácie).

V prípade problémov s aplikáciou (zacyklenie sa, nečakané ukončenie aplikácie, hľadanie chýb, sledovanie krokov aplikácie, ...) sa môže zapnúť okno s krokovaním.

[1] konštanta, ktorá určuje vzdialenosť medzi bodmi na výstupe A* algoritmu je v aplikácií voliteľná (v aplikačnom konfiguračnom súbore). Menšia vzdialenosť má za následok detailnejšiu cestu no za cenu vyšších časových nárokov - musí sa preveriť viac bodov, kadiaľ cesta mohla pokračovať.

Krokovanie sa rozumie zoznam o ukončení základných procesov spolu s časom ukončenia, daného procesu. Krokovanie algoritmu sa otvorí v novom samostatnom okne. Príklad krokov :

- Začatie počítania nového koridoru
- Vypočitanie podkoridorov
- Rozšírenie podkoridorov
- Vypočítanie alfa hrán

Kvôli testovaniu a meraniu vlastností kreslenia a najmä kvôli modelom zachovávania mentálnej mapy hrán je pripravený špeciálny nástroj - "robot". Po spustení tohto "robota"sa zamkne ovládanie aplikácie, graf sa nastaví do východzej štruktúry (ako pri zapnutí aplikácie) a vykoná sa niekoľko interakcií s grafom. Po ukončení týchto prednastavených automatických interakcií sa znova umožni práca s aplikáciou. Týmto spôsobom získali namerané hodnoty (kapitola 4) a vylaďovali aj samotné modely zachovávania mentálnej mapy hrán.

Záver

Interakcia s grafom a vykresľovanie grafu môže spôsobiť pri snahe zachovať estetické kritéria kritické zmeny vo vykreslení. V pripravenej aplikácií sme ukázali, ako jednoduchý algoritmus (A*), môže vytvoriť veľmi odlišné vykreslenie kvôli malej interakcií s grafom. Kresliaci algoritmus v aplikácií hľadá cesty, ktoré nepretínajú žiadny vrchol. Toto jednoduché statické kritérium spôsobilo pri niektorých interakciach veľké zmeny vo vykreslení. Pre ovládanie týchto zmien, nezávisle nad zvoleným kresliacim algoritmom, sme navrhli model mentálnej mapy hrán a dva postupy, ako ju zachovávať.

Model mentálnej mapy pre cestu sme zadefinovali ako postupnosť úsečiek. Ukázali sme dva postupy, ako takúto mentálnu mapu zachovávať.

Jednoduchší prvý model zachovávania mentálnej mapy pristupoval k problému priamočiaro. Dobré výsledky (ktoré dosahoval tento model pri meraniach) boli často dosiahnuté za cenu porušenia statických kresliacich kritérií či neprijateľného vykreslenia cesty.

Druhý model zachovávania mentálnej mapy simuluje intuitívny prístup k riešeniu problému - vytvárať škálovateľnú oblasť, v ktorej dovolíme zmeny cesty. Túto oblasť sme nazvali koridor. Ukázali sme, ako rozdeliť problém hľadania takéhoto koridoru na menšie podproblémy (koridor sa skladá z jednoduchších podkoridorov) a následne aj vyriešili spájanie vyriešených podproblémov (do výsledného koridoru).

Spájanie podkoridorov a hľadanie oblastí, ktoré čo najdetailnejšie popisujú množinu bodov, sme realizovali pomocou alfa hrán. Výsledne alfa hrany sa ale museli ďalej spracovať - odstrániť nepotrebné hrany, overovať či z výstupných alfa hrán vieme získať koridor atď. Výstupné alfa hrany sa preto intuitívne transformovali na graf a pomocou hľadania ťahu v grafe sa overoval a zostavoval koridor. V prípade neúspešného nájdenia koridoru (väčšinou kvôli zle nastavenej alfa hodnote) sa hľadanie opakuje s novou alfa hodnotou.

Pre overenie našich domienok o zlepšení vykresľovania grafov pri zachovávaní mentálnej mapy hrán sme navrhli tri spôsoby / hodnotenia, ako túto mentálnu mapu

udržiavať počas interakcie. Pri zachovávaní mentálnej mapy pomocou koridorov sme namerali mierne zlepšenie kvality vo vykreslení v dvoch spôsoboch meraní. Pri treťom bolo meranie takmer identické, ako keby sa mentálna mapa neudržiavala.

Literatúra

[1] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis (1998), Graph Drawing: Algorithms for the Visualization of Graphs (1st ed.). Prentice Hall PTR, Upper Saddle River, NJ, USA.

[2] F J Brandenburg (1996), Graph drawing : symposium on graph drawing, Springer, cop.

[3] T. and Rahman, M.S (2004), Planar graph drawing.Lecture Notes Series on Computing Series. Nishizeki, World Scientific Publishing Company, Incorporated

[4] William T. Tutte (1963), How to draw a graph. Proc. London Math. Society,13(52):743-768

[5] Peter Eades edited by D. S. Meek, van G. H. J. Rees (1984), A Heuristic for Graph Drawing Congressus Numerantium, Vol. 42, pp. 149-160

[6] R. Hadany and D. Harel (1999), A Multi-Scale Algorithm for Drawing Graphs Nicely. Technical Report. Weizmann Science Press of Israel, Jerusalem, Israel.

[7] David Harel and Yehuda Koren (2000), A fast multi-scale method for drawing large graphs. In Proceedings of the working conference on Advanced visual interfaces (AVI '00). ACM, New York, NY, USA, 282-285

[8] T. Munzner (1997) H3: laying out large directed graphs in 3D hyperbolic space, In Proceedings of the IEEE Symposium on Information Visualization (InfoVis '97) (INFOVIS '97). IEEE Computer Society, Washington, DC, USA.

[9] Stephen G. Kobourov and Kevin Wampler (2005), Non-Euclidean Spring Embedders. IEEE Transactions on Visualization and Computer Graphics 11, 757-767

[10] Albert-Laszlo Barabasi. 2003. Linked: How Everything is Connected to Everything Else and what it Means. Plume

[11] Yi-Yi Lee, Chun-Cheng Lin, and Hsu-Chun Yen (2006), Mental map preserving graph drawing using simulated annealing. In Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation - Volume 60 (APVis '06), Kazuo Misue, Kozo Sugiyama, and Jiro Tanaka (Eds.), Vol. 60. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 179-188.

[12] Peter Saffrey and Helen Purchase (2008), The "mental map"versus "static aesthetic"compromise in dynamic graphs: a user study. In Proceedings of the ninth conference on Australasian user interface - Volume 76 (AUIC '08), Vol. 76. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 85-93.

[13] R. A. Jarvis (1973), On the Identification of the Convex Hull of a Finite Set of Points in the Plane. Inf. Process. Lett. 2(1): 18-21

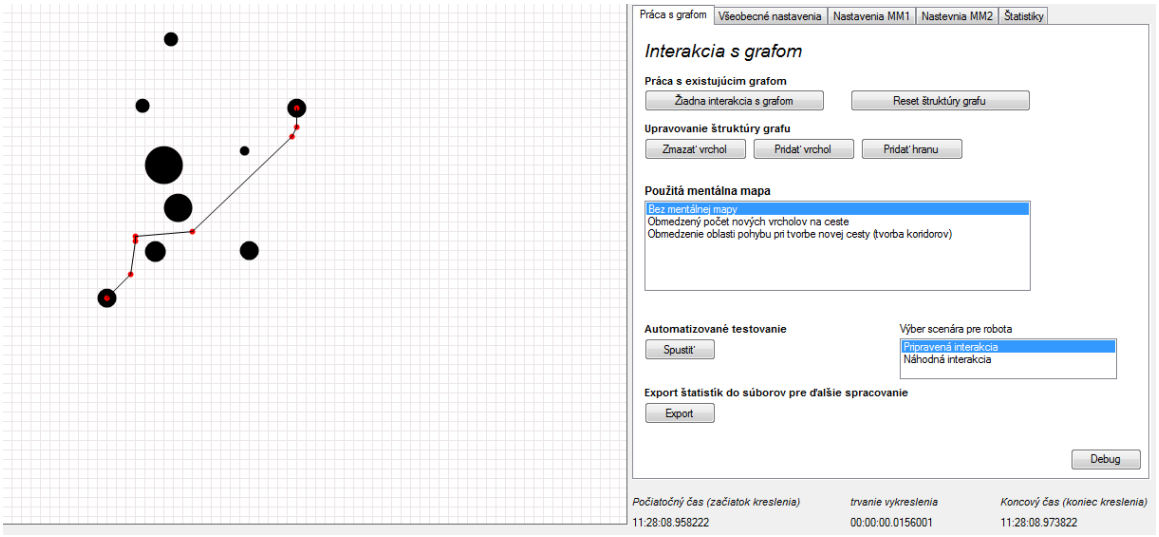
[14] Jin-Seo Park, Se-Jong Oh (2012), A New Concave Hull Algorithm and Concaveness Measure for n-dimensional Datasets. J. Inf. Sci. Eng. 28(3): 587-600

[15] B. Delaunay: Sur la sphère vide, Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 7:793–800, 1934

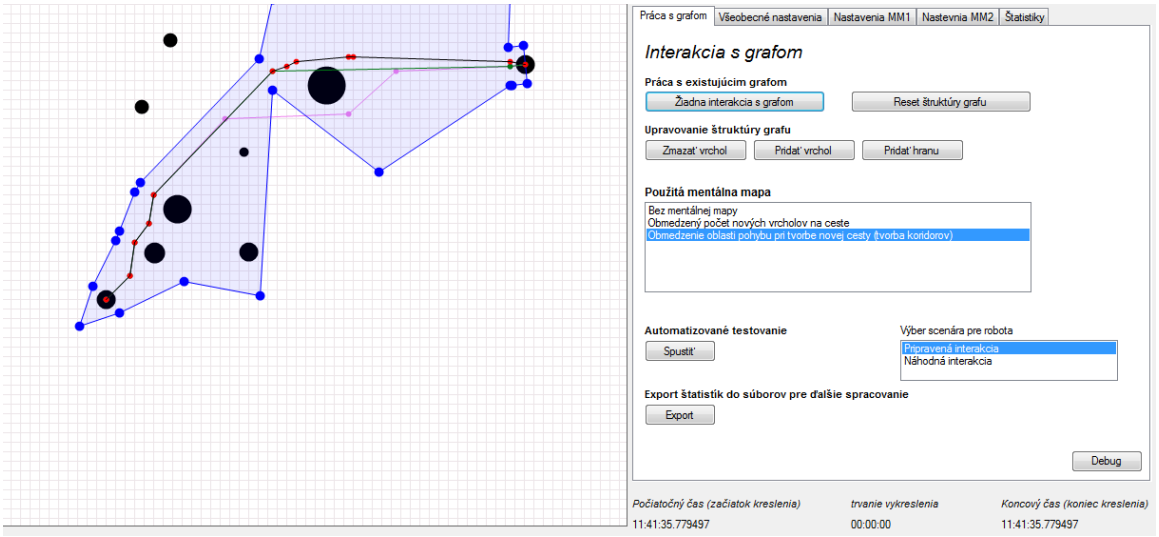
[16] Franz Aurenhammer (1991), Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Comput. Surv. 23, 345-405

[17] Edelsbrunner, Herbert; Kirkpatrick, David G.; Seidel, Raimund (1983), On the shape of a set of points in the plane, IEEE Transactions on Information Theory 29 (4): 551–559

Prílohy



Obr. 5.1: *Náhľad aplikácie po spustení.*



Obr. 5.2: *Príklad vykresleného grafu s koridorom.*